AD-A053 780 ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) F/G 9/2
THE DEVELOPMENT OF QUASI-NEWTON METHODS FOR UNCONSTRAINED MINIM--ETC(U)
AUG 77 A G PURCELL

UNCLASSIFIED RAE-TR-77132 DRIC-BR-60413 NL

| OF |
AD
A053780

END
DATE
FILMED
6 -78
DDC

UNLIMITED

TR 77132

BR60413

(14) RAE-TR-77132

ROYAL AIRCRAFT ESTABLISHMENT

*

(9) Technical Report 77132

(11) August 1977

(12) 70 p.

(6) # THE DEVELOPMENT OF QUASI-NEWTON METHODS FOR UNCONSTRAINED MINIMISATION.

by

(10) A.G. Purcell

(18) DRIC

(19) BR-60413

*

Procurement Executive, Ministry of Defence
Farnborough, Hants

UNLIMITED

310 450 JOB

UDC 517.97 : 512.643

ROYAL AIRCRAFT ESTABLISHMENT

Technical Report 77132

Received for printing 23 February 1978

THE DEVELOPMENT OF QUASI-NEWTON METHODS FOR UNCONSTRAINED MINIMISATION

by

A. G. Purcell

ADDENDUM

The following remark should be added on page 44:-

Enquiries on the purchase or use of NPL routines for Function Minimisation should be addressed to NPL.

UDC 517.97 : 512.643

ROYAL AIRCRAFT ESTABLISHMENT

Technical Report 77132

Received for printing 30 August 1977

THE DEVELOPMENT OF QUASI-NEWTON METHODS FOR UNCONSTRAINED MINIMISATION

by

A. G. Purcell

SUMMARY

Formulae for updating matrices in connexion with the quasi-Newton iteration are derived so as to emphasise the principles involved. Computational aspects are discussed and two FORTRAN programs for nonlinear minimisation subject to bounds on the variables are described and their use of finite difference derivatives, treatment of bounds, line searches and post-optimal sensitivity facilities are compared to demonstrate the manner in which the subject has progressed in recent years. Brief user guides to the two programs are contained in Appendices.

Departmental Reference: Math-Comp 229

LIST OF CONTENTS

1 INTRODUCTION

Many engineering problems can be expressed mathematically as the optimisation (meaning maximisation or minimisation) of a function of several variables, usually subject to constraints which may range in complexity from simple upper and/or lower bounds on the variables to highly nonlinear functions whose partial derivatives (with respect to the variables) cannot be obtained explicitly.

Applied Mathematics Division at RAE has followed developments in nonlinear optimisation, largely through Piggott, ever since the subject began to gather momentum in the mid-1960s. A succession of very useful computer programs was produced implementing the leading methods. The present work is a result of our continuing interest and reflects the progress made in unconstrained minimisation since the last Report (Piggott[1]).

The program M21UNC which we shall be describing hopefully maintains the standard of reliability set by Piggott while being significantly more efficient than its predecessor, M0402. The latter is also discussed for comparison purposes and because M21UNC contains subroutines written at NPL thereby restricting it to applications within RAE. Both programs are quite general and thus should be of interest to anyone in RAE with a problem in optimisation (including nonlinear least squares curve fitting and even the solution of nonlinear equations). They also form the basis of two programs for solving nonlinearly constrained minimisation problems (Purcell[2]).

It is convenient to incorporate bounds on the variables into an 'unconstrained' routine since they are a help rather than a hindrance in that they effectively reduce the dimensionality of the unconstrained problem. Thus in what follows, when referring to unconstrained problems we include the possibility of bounds and reserve the term 'constraints' for more complicated restrictions.

Many copies of M0402 have been supplied to RAE users. Applications (mainly least squares curve fitting) have included wind-tunnel modelling, design of digital filters, helicopter vibration, satellite launching, modelling human response times to visual data, etc. M21UNC, being a new program, has not been used in earnest in its own right but its robustness and performance have
132 been verified on constrained optimisation problems originating in Aerodynamics Department.

Both M0402 and M21UNC employ a quasi-Newton algorithm. This has been the most successful approach to nonlinear minimisation but every aspect of the

original algorithm (Fletcher and Powell[3]) has been revised and improved. The dual purposes of this Report are to explain how Newton's iteration was developed into a practical algorithm, with particular emphasis on matrix recurrence formulae, and to introduce a FORTRAN computer program which incorporates all important subsequent enhancements.

The rest of this Report is laid out as follows. We begin by describing Newton's iteration for function minimisation and explain what is meant by 'quasi-Newton'. In section 3 we derive formulae for updating the approximate inverse Hessian matrix, $H^{(i)}$ to $H^{(i+1)}$ where i is the iteration number and in so doing we hope that the underlying principles will become clear. By contrast, most previous authors have tended to state their particular formula first and demonstrated its properties afterwards. We shall show that the most important (*ie* the most widely used) updates are also the simplest which fulfil the fundamental requirements. The way in which these specific formulae fit into general families is explained and equivalent expressions for correcting approximations to the Hessian itself ($B^{(i)}$ to $B^{(i+1)}$) are deduced and their advantages outlined.

Section 4 considers the difficulties arising when partial derivatives have to be approximated by finite differences because the objective function $F(\underline{x})$, is too complicated to differentiate explicitly. The ways in which bounds on the variables can be incorporated are discussed in section 5 while section 6 explains the one-dimensional minimisation (line search) procedure. Section 7 summarises the relative merits of M0402 and M21UNC and section 8 deals with sensitivity analysis. There we describe how the program user can and should (unless he is only interested in reducing F to some 'acceptable' level) explore the accuracy of the solution found and its sensitivity to small changes in the variables and/or the calculation of F. The algorithms implemented as M0402 and M21UNC are sketched at the end of section 3.3 and as section 5.3 respectively while user guides to the two programs appear as Appendices B and C.

2 THE QUASI-NEWTON ITERATION

This Report is concerned with the following problem.

$$\text{Minimise } F(x_1, x_2, \ldots, x_k, \ldots, x_n) \qquad (2\text{-}1)$$

subject to $XL_k < x_k < XU_k \qquad k = 1, 2, \ldots, n$

where F is a nonlinear function of the x_k. The simplest situation would be if F was a quadratic form and the method we shall describe is based on quadratic

approximation of general objective functions. Most of the effort is therefore expended on directing the solution path into a neighbourhood of a minimum where F is effectively quadratic so that rapid final convergence is achieved. We must emphasise that the global minimum will only be obtained if the problem is convex. Furthermore, the algorithm theoretically requires the objective function and its first and second derivatives to be continuous. For many practical problems these are severe conditions. However, the method can often be successfully applied under weaker conditions or, at worst, substantial progress made towards a solution.

The method consists of a sequence of steps (iterations) given by

$$\underline{x}^{(i+1)} = \underline{x}^{(i)} - \alpha^{(i)} H^{(i)} \underline{g}^{(i)} \qquad (2\text{-}2)$$

where $\underline{g}^{(i)}$ is the vector of first partial derivatives at $\underline{x}^{(i)}$. This is merely a combination of Newton's iteration for locating a zero of a function and the condition for a turning point ($g = 0$) extended to several variables. In one variable

$$\left(\frac{dF}{dx}\right)^{(i+1)} = \left(\frac{dF}{dx}\right)^{(i)} + (x^{(i+1)} - x^{(i)}) \frac{d^2F}{dx^2} + \ldots = 0 .$$

Therefore,

$$x^{(i+1)} = x^{(i)} - \left(\frac{dF/dx}{d^2F/dx^2}\right)^{(i)} . \qquad (2\text{-}3)$$

Iteration (2-3) reaches the turning point in one step if F is quadratic and the point is a minimum if $d^2F/dx^2 > 0$. In the multi-dimensional analogue (2-2), $H^{(i)}$ is the inverse of the second derivative (Hessian) matrix and it must be positive definite at a minimum, *ie*

$$\underline{u}^T H^* \underline{u} > 0 \qquad \text{for any non-zero } \underline{u} . \qquad (2\text{-}4)$$

The fundamental idea of the quasi-Newton iteration (Davidon[4]) is that H should never be evaluated explicitly by double differentiation and matrix inversion. Instead H is gradually constructed by incorporating information acquired at each iteration (specifically, $\underline{g}^{(i+1)} - \underline{g}^{(i)}$ and $\underline{x}^{(i+1)} - \underline{x}^{(i)}$). In principle then, H is updated after every iteration until at the solution, x^*, where $\underline{g}^* = \underline{0}$, H^* is the exact inverse of the true Hessian, $G(\underline{x}^*)$.

An alternative interpretation of (2-2) is to treat $(\underline{x}^{(i+1)} - \underline{x}^{(i)})$ as the direction of steepest descent at $\underline{x}^{(i)}$ with respect to the metric $H^{(i)}$. Thus if $H = I$, the unit matrix, we would have the familiar first order steepest descent iteration while $H = G^{-1}$ would be second order steepest descent (if G is positive definite). Because H changes from one iteration to the next, the method was also termed 'variable metric'.

Now it is well known that Newton's iteration (2-3) can be unstable even in one dimension so the parameter α is introduced to prevent divergence. The vector $H\underline{g}$ is now regarded as a direction with α determining how far along it to go before re-evaluating $\underline{g}$ and updating H. It is therefore expected that as a minimum is approached F will become locally quadratic, $H \to G^{-1}$ and $\alpha \to 1$. It should already be apparent that the efficiency of the process which determines each $\alpha^{(i)}$ will be a major factor affecting the efficiency of the algorithm as a whole. Likewise the procedure for improving H will have to guard against ill-conditioning of the resulting matrix.

In the next section we consider methods by which H may be updated with the emphasis initially on generating the (constant) inverse Hessian of a quadratic form in a finite number of steps. Simultaneously we will discover practical criteria for determining α and tests to protect the conditioning of H many of which are immediately applicable to more general objective functions.

3 MATRIX UPDATING FORMULAE

As implied in the previous section, the theory behind the formulae for updating H is only strictly applicable to quadratic objective functions. It is hoped that as the minimum of a non-quadratic function is approached this approximation will become increasingly accurate and that ultimately, quadratic convergence will be achieved. Thus in what follows we shall assume a purely quadratic F in which case the true Hessian matrix, G, does not change with $\underline{x}$. The problem is then to construct G or G^{-1} using only $\underline{x}$ and $\underline{g}(\underline{x})$. ($F(\underline{x})$ is not needed for this purpose.) As we proceed we will remind ourselves periodically of our ultimate goal; to develop an algorithm which can safely and efficiently be applied to general objective functions.

3.1 Solving quadratic problems without matrix updating

Beginning at $\underline{x}^{(0)}$ where the partial derivative vector is $\underline{g}^{(0)}$ we step to other points $\underline{x}^{(1)}, \ldots \underline{x}^{(i)}, \ldots$ evaluating $\underline{g}^{(1)}, \ldots \underline{g}^{(i)}, \ldots$ as we go. Because F is quadratic we have

$$\underline{g}^{(i+1)} = \underline{g}^{(i)} + G(\underline{x}^{(i+1)} - \underline{x}^{(i)}) \ . \qquad (3\text{-}1)$$

Writing

$$\underline{q}^{(i)} = \underline{g}^{(i+1)} - \underline{g}^{(i)} \ ; \quad \underline{p}^{(i)} = \underline{x}^{(i+1)} - \underline{x}^{(i)} \qquad (3\text{-}2)$$

this becomes

$$\underline{q}^{(i)} = G\underline{p}^{(i)} \ . \qquad (3\text{-}3)$$

It is then intuitively obvious that by generating n vectors $\underline{p}^{(i)}$ we would be able to determine G from

$$Q = GP \qquad (3\text{-}4)$$

where the columns of Q,P are the vectors $\underline{q}^{(i)}$, $\underline{p}^{(i)}$. Of course, the $\underline{p}^{(i)}$ must be linearly independent so that P can be inverted. The solution to our minimisation problem is then obtained in one further step, given by (3-1),

$$\underline{g}^{(n+1)} = \underline{g}^{(n)} + G(\underline{x}^{(n+1)} - \underline{x}^{(n)}) = \underline{0} \ .$$

Therefore

$$\underline{x}^* = \underline{x}^{(n+1)} = \underline{x}^{(n)} - G^{-1}\underline{g}^{(n)} \ . \qquad (3\text{-}5)$$

From here on we rely on the reader to remember that x, g, p, q are vectors. New vectors (*eg* $\underline{d}$, $\underline{y}$, $\underline{z}$) will be underlined when they first appear but not subsequently.

So what has been achieved so far? We have not mentioned the notion of approximating G (or G^{-1}) and updating it after each iteration. We have merely shown that knowledge of the gradient vector at (n + 1) suitably chosen points is sufficient to completely determine the second derivative matrix of a quadratic function. By 'suitably chosen' we mean that the vectors $(\underline{x}^{(i+1)} - \underline{x}^{(i)})$ must be linearly independent. It is easy enough to predetermine a satisfactory set of points and, in particular, the simplest and most obvious choice is to make P the unit matrix by stepping along the coordinate axes in turn, *ie*

132

$$x_k^{(i)} = 0 \quad k > i$$
$$= 1 \quad k \leqslant i \qquad i = 0, 1 \ldots n \ .$$

Hence $Q = G$ but a matrix inversion is still necessary in order to make the final step to $\underline{x}^*$.

3.2 Rank-1 updating and positive definiteness

The above procedure is not very practical when the objective function is non-quadratic because G varies with $\underline{x}$. A more general strategy for choosing the $\underline{p}^{(i)}$ is needed while retaining the linear independence property on quadratic F . The important contribution made by Davidon[4] and developed by Fletcher and Powell[3] was to recognise that the Newton iteration (2-2) could be used and to deduce automatic methods for generating $H^{(i)}$ and choosing $\alpha^{(i)}$. In particular, Davidon announced a formula by which $H^{(i)}$ could be updated once $\alpha^{(i)}$, $\underline{x}^{(i+1)}$ and $\underline{g}^{(i+1)}$ have been evaluated.

Two important properties of the Hessian matrix (or inverse) of a general function are symmetry and, at a minimum, positive definiteness (*ie* $\underline{u}^T H^* \underline{u} > 0$ for any non-zero vector $\underline{u}$). We will return to the second of these later. Meanwhile, the first property leads us to investigate the simplest possible symmetry preserving update, the addition of a symmetric matrix of rank-1.

$$H^{(i+1)} = H^{(i)} + a^{(i)} \underline{z}^{(i)} \underline{z}^{(i)T} . \qquad (3\text{-}6)$$

For quadratic F we know we can deduce the correct Hessian in n steps. It is desirable that a strategy which amends an approximation should do likewise, *ie*

$$H^{(n)} G = I .$$

Now if we can find n linearly independent vectors $\underline{v}^{(j)}$, $j = 0,1,\ldots(n-1)$ each satisfying

$$H^{(n)} G \underline{v}^{(j)} = \underline{v}^{(j)} \qquad (3\text{-}7)$$

then we can be sure that $H^{(n)} = G^{-1}$ because only the unit matrix has all unit eigenvalues. A natural choice for $\underline{v}^{(j)}$ is $\underline{p}^{(j)}$ and so with (3-3), (3-7) becomes

$$H^{(n)} q^{(j)} = p^{(j)} \qquad j = 0,1\ldots(n-1) . \qquad (3\text{-}8)$$

Hence if we make $H^{(i+1)}$ satisfy

$$H^{(i+1)} q^{(j)} = p^{(j)} \qquad j \leq i \qquad (3\text{-}9)$$

then $H^{(n)} = G^{-1}$ is guaranteed and the minimum, x^* will be reached on the next iteration. Note that unlike (3-5) this strategy does not involve any matrix inversions.

It happens that imposing requirement (3-9) for $j = i$ on equation (3-6) gives a unique formula and that (3-9) is thereby simultaneously satisfied for $j < i$.

$$H^{(i+1)}q^{(i)} = p^{(i)} = H^{(i)}q^{(i)} + a^{(i)}z^{(i)}z^{(i)T}q^{(i)} \quad . \tag{3-10}$$

It can easily be shown that the only symmetric rank-1 update is characterised by

$$\left.\begin{aligned} z^{(i)} &= p^{(i)} - H^{(i)}q^{(i)} \\ a^{(i)} &= \frac{1}{q^{(i)T}z^{(i)}} \end{aligned}\right\} \tag{3-11}$$

and (by induction) that relations (3-9) automatically hold for $j < i$ (see Luenberger[5], pp 193-194). Furthermore, Murtagh and Sargent[6] showed that the quasi-Newton iteration ($p^{(i)} = - H^{(i)}g^{(i)}$) generates directions which are linearly independent if conditions (3-9) are satisfied. Thus for quadratic functions we have a virtually complete algorithm, (A-1), as follows:

(1) Choose $x^{(0)}$, $H^{(0)}$ (= I, say). Evaluate $g^{(0)}$. Set $i = 0$.

(2) Calculate $p^{(i)}$. Hence $x^{(i+1)} = x^{(i)} + p^{(i)}$.
Note that $p^{(0)}$ to $p^{(n-1)}$ could still be pre-determined while $p^{(n)} = - H^{(n)}g^{(n)}$.

(3) If $i = n$, $x^{(i+1)} = x^*$ so stop, otherwise

(4) Evaluate $g^{(i+1)}$. Hence $q^{(i)}$.

(5) Update $H^{(i)}$ to $H^{(i+1)}$ using (3-6) and (3-11).

(6) Set $i = i + 1$; go to (2).

This algorithm is not practical even on quadratic functions because the conditioning of $H^{(i+1)}$ has not been monitored. We mentioned above that H^* is positive definite so it would seem a worthwhile precaution to start with $H^{(0)}$ positive definite and to maintain $H^{(i)}$ so subsequently. A weakness of the rank-1 formula is thereby exposed; it cannot guarantee $H^{(i+1)}$ positive definite even on quadratic functions. However, Murtagh and Sargent[6] noted that G^{-1} would still be obtained if n updates could eventually be performed, *ie* the n linearly independent directions need not be generated consecutively.

132

An obvious sufficient condition for $H^{(i+1)}$ to be positive definite is $a^{(i)} > 0$ in (3-6), *ie* from (3-11)

$$q^T(p - Hq) > 0 \ . \tag{3-12}$$

Otherwise we set $H^{(i+1)} = H^{(i)}$ and continue. Unfortunately there is no guarantee that inequality (3-12) will be satisfied n times even if F is quadratic (*eg* Luenberger[5], p 214). A further sufficient condition for $H^{(i+1)}$ to be positive definite was demonstrated by Murtagh and Sargent[6]

$$ag^T z < 0 \ . \tag{3-13}$$

We can combine these conditions so that $H^{(i)}$ is updated if

$$\text{(a)} \quad g^{(i+1)T}(p - Hq) > g^{(i)T}(p - Hq)$$

or (3-14)

$$\text{(b)} \quad g^{(i)T}(p - Hq) > 0 \ .$$

To summarise, the great virtue of the rank-1 formula is that, on quadratic functions, n repetitions of iteration (2-2) with $\alpha = 1$ will potentially completely determine the Hessian matrix. The over-riding disadvantage is that H may become ill-conditioned and in attempting to prevent this happening we risk the algorithm failing to achieve $H \to G^{-1}$ at all. We are not dismayed by losing n-step convergence but the possibility of complete failure is intolerable. Of course, it is quite likely that a path to the solution which maintains H positive definite goes a long way round but, in addition to enabling the conditioning of H to be controlled, such a route has a feature which is reassuring on general functions, namely that a decrease in F can be obtained at each step. This follows from the Taylor expansion,

$$F(\underline{x} + \underline{p}) = F(\underline{x}) + p^T g + \tfrac{1}{2} p^T G p + \ldots \ ,$$

but

$$p = - \alpha H g \ . \tag{3-15}$$

Therefore $F^{(i+1)} - F^{(i)} = - \alpha g^T H g + \tfrac{1}{2}\alpha^2 g^T H G H g + \ldots$ and by choosing α sufficiently small and positive a decrease in F is assured. All quasi-Newton algorithms for unconstrained minimisation strive to keep H positive definite. Likewise, practical algorithms have in common that rank-2 updating formula are used in preference to rank-1, initially because a strategy was available which

ensured that H could be modified in safety on every iteration. We now turn our attention to rank-2 modification rules and confirm their various properties.

3.3 Rank-2 updates

The general form is

$$H^{(i+1)} = H^{(i)} + a^{(i)}\underline{z}^{(i)}\underline{z}^{(i)T} + b^{(i)}\underline{y}^{(i)}\underline{y}^{(i)T} \qquad (3\text{-}16)$$

and there is now an infinity of solutions which satisfy the conditions (3-9) so it is possible to specify additional desirable features and/or to consider families of rank-2 formulae. In particular, if we could select a set of n directions $\underline{d}^{(i)}$ which were mutually conjugate with respect to the Hessian, G, of a quadratic function then not only would $H^{(n)} = G^{-1}$ but also $\underline{x}^{(n)} = d\underline{x}^*$ and an iteration would have been saved. From here on $\underline{d}^{(i)}$ represents direction of search while $\underline{p}^{(i)}$ is the actual step taken, *ie*

$$\underline{p}^{(i)} = \underline{x}^{(i+1)} - \underline{x}^{(i)} = \alpha^{(i)}\underline{d}^{(i)} . \qquad (3\text{-}17)$$

The conjugacy condition is therefore

$$\underline{d}^{(i)T}G\underline{d}^{(j)} = 0 \qquad j \neq i$$

or

$$\underline{p}^{(i)T}G\underline{p}^{(j)} = 0 \qquad j \neq i . \qquad (3\text{-}18)$$

With (3-3)

$$p^{(i)T}q^{(j)} = 0 \qquad j \neq i . \qquad (3\text{-}19)$$

In particular, from (3-15)

$$p^{(i+1)T}q^{(i)} = -\alpha^{(i+1)}g^{(i+1)T}H^{(i+1)}q^{(i)}$$

$$= -\alpha^{(i+1)}g^{(i+1)T}p^{(i)} \qquad \text{from (3-9).}$$

Thus, the conjugacy condition becomes

$$g^{(i+1)T}d^{(i)} = 0 \qquad (3\text{-}20)$$

which is purely the condition that the function F be minimised along the

direction $\underline{d}^{(i)}$. This is a key observation and suggests the following practical algorithm (A-2)

(1) $d^{(i)} = - H^{(i)} g^{(i)}$. (3-15a)

(2) Find $\alpha^{(i)}$ such that $x^{(i)} + \alpha^{(i)} d^{(i)}$ minimises F along the direction $d^{(i)}$. Hence get $g^{(i+1)}$, $q^{(i)}$.

(3) Update H to satisfy (3-9).

(4) Set $i = i + 1$; go to (1).

The process of selecting $\alpha^{(i)}$ is called a 'line search' and the term 'exact line search' denotes that (3-20) is satisfied.

We now show that (3-9), (3-15a) and (3-20) together guarantee (3-19). The proof is inductive. First we observe from (3-18) that only $j < i$ need be considered. Therefore we have to prove that

$$d^{(i+1)T} q^{(j)} = 0 \qquad j < i + 1 \quad (3\text{-}21)$$

assuming that

$$p^{(i)T} q^{(j)} = d^{(i)T} q^{(j)} = 0 \qquad \text{for } j < i .$$

We have already noted that (3-20) establishes (3-21) for $j = i$ so only $j < i$ remain to be proved.

Now, using (3-15a) and the definition of q ((3-2))

$$d^{(i+1)} = - H^{(i+1)} (g^{(i)} + q^{(i)})$$

$$= - H^{(i+1)} g^{(i)} - p^{(i)} \qquad \text{from (3-9)}$$

ie

$$d^{(i+1)T} q^{(j)} = - g^{(i)T} H^{(i+1)} q^{(j)} - p^{(i)T} q^{(j)} .$$

The second term is zero by assumption, while from (3-9)

$$g^{(i)T} H^{(i+1)} q^{(j)} = g^{(i)T} p^{(j)} = g^{(i)T} H^{(i)} q^{(j)} \qquad j < i$$

ie

$$d^{(i+1)T} q^{(j)} = d^{(i)T} q^{(j)} \qquad \text{from (3-15a)}$$

$$= 0 \qquad \text{by assumption when } j < i .$$

To complete the proof we observe that $p^{(1)T}q^{(0)} = 0$ as a consequence of the first exact line search.

Thus with a quadratic objective function algorithm (A-2) will generate a sequence of vectors $\underline{d}^{(i)}$ which are mutually conjugate with respect to the true Hessian (rather than just being linearly independent) and the solution $\underline{x}^*$ will be obtained in, at most, n steps providing that the approximation H to G^{-1} is updated after each iteration. This becomes quite clear when we use (3-15a) to express the conjugacy relations as

$$g^{(i)T}H^{(i)}q^{(j)} = 0 \qquad j < i$$

and by (3-9)

$$g^{(i)T}p^{(j)} = 0 \qquad j < i \ . \qquad (3\text{-}22)$$

Thus each $g^{(i)}$ is orthogonal to all the previous steps. After n linearly independent iterations $p^{(0)}, \ldots . p^{(n-1)}$ we must have $\underline{g}^{(n)} = \underline{0}$.

As yet, we have not implied any particular method of updating $H^{(i)}$ except to insist that relations (3-9) are satisfied. Hence even the rank-1 formula could be used although there is no virtue in so doing because (3-20) is insufficient to guarantee that the matrix update can proceed. However, we shall show presently that (3-20) is sufficient to ensure that $H^{(i+1)}$ is positive definite when a formula of rank-2 is employed even on non-quadratic functions.

First let us derive a simple rank-2 update; taking $j < i$, (3-16) and (3-9) give

$$H^{(i+1)}q^{(j)} = H^{(i)}q^{(j)} + a^{(i)}z^{(i)}z^{(i)T}q^{(j)} + b^{(i)}y^{(i)}y^{(i)T}q^{(j)}$$

$$= p^{(j)} \ .$$

Therefore $$a^{(i)}z^{(i)}z^{(i)T}q^{(j)} + b^{(i)}y^{(i)}y^{(i)T}q^{(j)} = 0 \ .$$

Now (3-19) shows immediately that $z^{(i)} = p^{(i)}$ would make one term zero while from (3-9) we can get

$$q^{(i)T}H^{(i)}q^{(j)} = q^{(i)T}p^{(j)} = 0 \qquad j < i$$

suggesting that $y^{(i)} = H^{(i)}q^{(i)}$ would conveniently zeroise the other term. Thus (3-16) becomes, dropping the superscript for clarity,

132

$$H^{(i+1)} = H + app^T + bHqq^TH \ .$$

a and b are determined by considering $j = i$ in (3-9).

$$H^{(i+1)}q = Hq + app^Tq + bHqq^THq = p \ .$$

Therefore, $$a = \frac{1}{p^Tq} \ ; \ b = -\frac{1}{q^THq} \ . \qquad (3\text{-}23)$$

These simple formulae for a, b, $\underline{y}$, $\underline{z}$ do indeed correspond to Davidon's[4] original proposals. Note that p and Hq have to be linearly independent for the update to be of rank-2. Note also that conditions (3-19) and (3-9) are now interlinked, with (3-20) providing the continuity. Thus if the line search is not exact then the next search direction will not be conjugate, the following update will not satisfy (3-9) and finite convergence is theoretically lost even on quadratic functions. However, an exact line search by computer is impossible anyway and is intuitively undesirable on non-quadratic functions where conjugacy and n-step convergence are irrelevant except perhaps very near the solution. Of more importance is that a criterion can be deduced for maintaining $H^{(i+1)}$ positive definite and that the exact line search strategy will cause this criterion to be automatically satisfied. Hence we can use the descent property of positive definiteness with inexact searches to manoeuvre into the (quadratic) vicinity of a minimum after which exact searches would guarantee convergence in $0(n)$ more iterations. We now derive the condition under which Davidon's rule will give $H^{(i+1)}$ positive definite if $H^{(i)}$ is .

The formula is

$$H^{(i+1)} = H + \frac{pp^T}{p^Tq} - \frac{Hqq^TH}{q^THq} \ . \qquad (3\text{-}24)$$

We need to show that $\underline{u}^TH^{(i+1)}\underline{u} > 0$ for arbitrary $\underline{u}$. Equation (3-24) gives

$$\text{Rhs} = u^THu + \frac{u^Tpp^Tu}{p^Tq} - \frac{u^THqq^THu}{q^THq} \ .$$

Since H is positive definite we can define a matrix R such that $R^TR = H$. Writing $\underline{\delta} = R\underline{u}$; $\underline{\varepsilon} = R\underline{q}$ gives

$$\text{Rhs} = \delta^T\delta + \frac{(u^Tp)(p^Tu)}{p^Tq} - \frac{(\delta^T\varepsilon)(\varepsilon^T\delta)}{\varepsilon^T\varepsilon}$$

$$= \frac{(u^Tp)^2}{p^Tq} + \frac{(\delta^T\delta)(\varepsilon^T\varepsilon) - (\delta^T\varepsilon)^2}{\varepsilon^T\varepsilon} \quad .$$

The second term is positive by the Cauchy-Schwartz inequality. Hence a sufficient condition for $H^{(i+1)}$ to be positive definite is

$$p^Tq > 0 \quad . \tag{3-25}$$

But

$$p^Tq = p^{(i)T}(g^{(i+1)} - g^{(i)}) .$$

With exact searches the first term is zero by definition while (3-15) gives

$$p^Tq = -p^Tg = \alpha g^THg > 0$$

because $\alpha > 0$ and H is positive definite.

Thus an exact line search will guarantee that Davidon's rule gives $H^{(i+1)}$ positive definite while (3-25) is the test which must be applied when the search is inexact. Note that condition (3-12) for the rank-1 formula is rather more demanding. However, the second test of (3-14) can be rewritten

$$g^T(p + Hg) > g^THg^{(i+1)}$$

ie

$$(1 - \alpha)g^THg > -g^{(i+1)T}d \quad .$$

If an exact search is performed then the condition is fulfilled if $\alpha < 1$. Of course there is no point in employing exact searches with the rank-1 formula anyway because its virtue lies in its potential to satisfy (3-9) by taking steps of arbitrary size when F is quadratic. On the other hand for quadratic problems (3-3) shows that $p^Tq > 0$ for any $\alpha \neq 0$ so the rank-2 update will succeed following steps of arbitrary size but convergence in n iterations is no longer assured.

This completes the basic theory of unconstrained optimisation algorithms. In particular, we have seen how Davidon's rank-2 formula arises naturally from the desire to generate a set of vectors mutually conjugate with respect to the unknown Hessian matrix, G of a quadratic function of n variables and how G can thereby be deduced in n iterations by varying the approximating metric, H. The exact line search is a consequence of this aim and is sufficient to ensure that H remains positive definite and hence that the objective function will decrease at every step. It should be stressed that while the symmetric rank-1 formula satisfying (3-9) is unique, Davidon's rank-2 update is but a simple example from an infinity of possibilities. In practice, line searches are never carried out exactly although it is intended that they will become more precise as a minimum is approached. Thus it becomes important to study the behaviour of rank-2 formulae when slack searches are used. Such work has led to the general adoption of the BFS rule (Broyden[7], Fletcher[8], Shanno[9] independently (1970)) which looks rather more complicated than (3-24).

$$H^{(i+1)} = H + \frac{(p - rHq)(p - rHq)^T}{rq^Tp} - \frac{Hqq^TH}{q^THq + q^Tp} \qquad (3\text{-}26)$$

where $$r = \frac{q^Tp}{q^Tp + q^THq} .$$

With exact searches it would generate the same sequence of points as (3-24) and it is subject to the same positive definite criterion ((3-25)).

Finally in this section we can state the algorithm (A-3) on which Mathematics and Computation Department FORTRAN subroutine M0402 is based.

(1) Set $i = 0$; choose $x^{(0)}$ and $H^{(0)}$ (= I normally) and evaluate $g^{(0)}$.

(2) Set $d^{(i)} = - H^{(i)}g^{(i)}$.

(3) Perform some sort of line search to get $\alpha^{(i)}$, $p^{(i)}$, $x^{(i+1)}$.

(4) Evaluate $g^{(i+1)}$, $q^{(i)}$. Obtain $H^{(i+1)}$ from (3-26) if $p^{(i)T}q^{(i)} > 0$. Otherwise set $H^{(i+1)} = H^{(i)}$.

(5) Terminate if the line search fails to make progress (or reset $H \to I$ and try to continue) or if the convergence criteria are satisfied. Otherwise set $i = i+1$ and go to (2).

132

Before discussing more recent algorithms we now look briefly at families of rank-2 matrix updating formulae.

3.4 Families of rank-2 updates

Davidon's formula corresponds to the specific selections $\underline{y} = H\underline{q}$; $\underline{z} = \underline{p}$ in (3-16) with a and b determined from conditions (3-9) with $j = i$. Clearly y and z could each be any linear combination of p and Hq

$$y = \phi p + Hq \; ; \; z = p + \psi Hq \tag{3-27}$$

and conditions (3-9) would automatically be satisfied for $j < i$ (for quadratic F and with exact line searches). The $j = i$ case can be identically fulfilled by (3-28) instead of (3-16)

$$H^{(i+1)} = H + \frac{pz^T}{z^T q} - \frac{Hqy^T}{y^T q} . \tag{3-28}$$

The combination of (3-27) and (3-28) gives the two parameter family first introduced by Huang[10]. To be strictly accurate, Huang's family has three parameters, the third one, ρ , multiplying the second term in (3-28). We have ignored it here because $H^{(i+1)}q^{(i)} = p^{(i)}$ is only obtained when $\rho = 1$. We mention it to indicate the type of variation which has been considered to try and improve the behaviour of algorithms away from the minimum of non-quadratic functions. For example Biggs[11] allowed ρ to vary on every iteration choosing it to in some sense take account of the non-quadraticity of F . Dixon[12] showed that on general functions all members of Huang's family having the same ρ generated the same sequence of points $\underline{x}^{(i)}$ providing the line search located the first minimum along $\underline{d}^{(i)}$ exactly. Therefore, the practical choice of ϕ and ψ in (3-27) will depend on the behaviour when slack searches are employed.

Now, it will be observed that (3-28) is not necessarily a symmetric update. If we impose the symmetry requirement then the family reduces to one free parameter. Putting (3-27) in (3-28), we find that the terms which are not automatically symmetric are

$$\frac{\psi pq^T H}{(p + \psi Hq)^T q} - \frac{\phi Hqp^T}{(\phi p + Hq)^T q} .$$

The necessary condition for symmetry is then

$$\psi(\phi p + Hq)^T q + \phi(p + \psi Hq)^T q = 0 ,$$

ie

$$\psi = \frac{-\phi p^T q}{(\phi p + (\phi + 1)Hq)^T q} . \qquad (3\text{-}29)$$

This resulting one parameter family of formulae (the symmetrised Huang family) is equivalent to Broyden's[13] family and we have

$$\phi = 0 \; ; \; \psi = 0 \qquad \text{Davidon's formula}$$

$$\phi = -1 ; \; \psi = -1 \qquad \text{Rank-1 formula}$$

$$\phi = \infty \; ; \; \psi = \frac{-p^T q}{p^T q + q^T Hq} \qquad \text{BFS formula} . \qquad (3\text{-}30)$$

Alternatively, when we combine (3-27) with (3-16) and determine a and b from $H^{(i+1)} q^{(i)} = p^{(i)}$ we get a two parameter family of symmetric rank-2 corrections:

$$H^{(i+1)} = H + \frac{(\phi + 1)zz^T}{(1 - \psi\phi)z^T q} - \frac{(\psi + 1)yy^T}{(1 - \psi\phi)y^T q} . \qquad (3\text{-}31)$$

Broyden's family is regained by setting $\phi = 0$, *ie*

$$H^{(i+1)} = H + \frac{(p + \psi Hq)(p + \psi Hq)^T}{(p + \psi Hq)^T q} - \frac{(\psi + 1)Hqq^T H}{q^T Hq} . \qquad (3\text{-}32)$$

By repeating the analysis following equation (3-24) it is easily shown that $H^{(i+1)}$ is necessarily positive definite if $H^{(i)}$ is, providing

$$p^T q > 0 \; ; \quad \frac{-p^T q}{q^T Hq} < \psi \leq 0 . \qquad (3\text{-}33a,b)$$

Thus Davidon's formula is at one end of the safe range while BFS is placed intermediately and the rank-1 update is indefinite (see (3-30)). Looked at in this way it might perhaps be suspected that BFS would be better behaved than Davidon.

Shanno[9] obtained the one parameter family from (3-31) by setting $\psi = 0$; $\phi + 1 = t$. He showed that the smallest eigenvalue of $H^{(i+1)}$ is maximised by the choice $t = \infty$ and that the spectral condition number of $H^{(i+1)}$ is then also favourable (Shanno and Kettler[14]). In the latter paper it was further argued that $t = \infty$ preserves certain eigenvalue properties during updating so that in effect the BFS correction could be deduced directly by recognising (3-34)

as a desirable property of $H^{(i+1)}$ and imposing it on the family (3-32) assuming exact line searches (*ie* (3-20)):

$$g^{(i+1)T}H^{(i+1)}g^{(i+1)} = g^{(i+1)T}H^{(i)}g^{(i+1)} \quad . \tag{3-34}$$

Of course, the theory is again only applicable to quadratic F .

The historical development of matrix modification schemes should by now be apparent. Davidon's formula (usually referred to as DFP to acknowledge its practical implementation by Fletcher and Powell[3]) was used for many years until it was realised that whole families of updating formulae existed. Meanwhile as the exact line search policy was relaxed, DFP tended to run into difficulties which were attributed to ill-conditioning of H . Although no really convincing theoretical results concerning inexact searches on non-quadratic functions have appeared, practical experience has confirmed that BFS is a superior tool. In the last five years the algorithm (A-3) has been revised to make it numerically more stable and more efficient in terms of function evaluations. The manner in which matrices are now recurred is briefly described in the next section.

3.5 More recent developments

It has been recommended, initially by Gill and Murray[15] that the Hessian itself be approximated rather than its inverse. It is interesting to follow through the developments of sections 3.2 and 3.3 in the revised form. Let $B^{(i)}$ be the approximation to G . The search direction is then obtained as the solution of

$$B^{(i)}\underline{d}^{(i)} = -\underline{g}^{(i)} \quad . \tag{3-35}$$

In practice B is stored in LDL^T form and the unit lower triangular matrix L and diagonal matrix D are updated separately. Any ill-conditioning of B tends to be reflected in D which can be rescaled when necessary to reduce its condition number. Any sparsity in B will show up in L whereas it would not in H but this is a feature which will not concern us; suffice it to say that the revised formulation is more adaptable to structured problems. It is also a relatively simple matter to ensure that $B^{(i+1)}$ is positive definite whereas test (3-25) does not guard against rounding error and there is no way of knowing what effect this has on $H^{(i+1)}$.

32

$$\underline{u}^T LDL^T \underline{u} > 0 \qquad \text{for arbitrary } \underline{u}(\neq \underline{0}) \ .$$

Defining $\underline{v} = L^T\underline{u}$ it is obvious that if D is positive definite (*ie* consists of all positive elements) then B will be also. However, it must be remembered that the conditioning of D is only a guide to that of B inasmuch as

$$\prod_{k=1}^{n} D_{kk} = \prod_{k=1}^{n} \lambda_{B_k} = \det(B) .$$

It is therefore quite easy to construct examples where all the D_{kk} are of similar magnitude but the eigenvalues of B, λ_{B_k}, differ vastly.

Formulae complementary to H-updates can be obtained by making the transformation

$$H \to B \; ; \quad p \to q \; ; \quad q \to p \; . \tag{3-36}$$

The unique rank-1 correction ((3-6) with (3-11)) is then

$$B^{(i+1)} = B + \frac{(q - Bp)(q - Bp)^T}{p^T(q - Bp)} \tag{3-37}$$

for which, of course, the property that if $H = B^{-1}$ then $H^{(i+1)} = B^{(i+1)}$ holds. This is not true of complementary rank-2 updates. An important feature of Hessian updating formulae is that matrix-vector products can be eliminated because

$$Hq \to Bp = -\alpha g$$

and the computation is more accurate as a result. An interesting feature of (3-37) is that the obvious positive definite preserving condition is

$$p^T(q - Bp) > 0 \tag{3-38}$$

which is not the same as the denominator condition for $H^{(i+1)}$ (3-12) but does correspond with (3-14b) which was effectively obtained by Murtagh and Sargent[6] but rather more laboriously

$$\begin{aligned} p^T(q - Bp) &= \alpha(d^Tq + p^Tg) \\ &= \alpha(-g^THq + p^Tg) \\ &= \alpha g^T(p - Hq) \; . \end{aligned}$$

Condition (3-12) must also be usable when B is recurred but it is less convenient to apply as $(B^{(i)})^{-1}g^{(i+1)}$ has to be specially computed.

An even more interesting revelation occurs when we apply (3-36) to the DFP rule (3-24). With (3-35) we find

$$B^{(i+1)} = B + \frac{qq^T}{p^Tq} + \frac{g^{(i)}g^{(i)T}}{d^Tg^{(i)}} \quad . \tag{3-39}$$

Now, if we apply Householder's[16] modification formula twice to (3-39) we will obtain the equivalent H-update. The result is (3-26), the BFS update, and not (3-24). Repeating briefly the analysis of section 3.3, the generalised rank-2 update is

$$B^{(i+1)} = B^{(i)} + c^{(i)}\underline{v}^{(i)}\underline{v}^{(i)T} + e^{(i)}\underline{w}^{(i)}\underline{w}^{(i)T} \tag{3-40}$$

and conditions (3-9) become

$$B^{(i+1)}p^{(j)} = q^{(j)} ; \qquad j < i \; . \tag{3-41}$$

Combining (3-40) and (3-41) gives

$$cvv^Tp^{(j)} + eww^Tp^{(j)} = 0 \qquad j < i \; .$$

The obvious vector choices (from (3-19) and (3-22)) are now

$$v = q \; ; \quad w = g^{(i)} \tag{3-42}$$

while the $j = i$ case gives

$$c = \frac{1}{p^Tq} \; ; \quad d = \frac{1}{d^Tg^{(i)}} \quad .$$

Thus, if the analysis had been carried out in the first place in a B-updating framework, the BFS formula would have been obtained as the simplest rank-2 update which generates conjugate directions with exact line searches and constructs the true Hessian of a quadratic function within n iterations. It therefore seems a rather curious coincidence that researchers apparently arrived at
132 the BFS formula for updating H by employing fairly sophisticated arguments (as indicated in section 3.4). Why should the 'best' formula so derived happen to be the most elementary solution of the complementary problem? It is all the

more remarkable because equation (3-26) was advocated for correcting H before Gill and Murray's[15] work with B-updates.

The new Mathematics and Computation Department FORTRAN program for unconstrained minimisation, M21UNC, uses LDL^T factorisation of the approximate Hessian, updating L and D in accordance with (3-39). The program becomes physically longer because

(a) the linear system $LDL^T\underline{d} = -g$ has to be solved for the search direction $\underline{d}$ and

(b) updating the factors L and D is not as straightforward as modifying H by (3-26). In addition care is taken to ensure that $B^{(i+1)}$ does not become ill-conditioned by rounding error. In fact, the manipulations involved in (3-26) are highly vulnerable to such inaccuracies but safeguards cannot be incorporated so readily. In practice, the appropriate parts of the H-update are performed in double precision arithmetic in M0402 to counter the effects of round-off.

Another idea which has been energetically pursued by one or two workers in recent years is updates which are self-scaling. Luenberger[5] observed that on quadratic functions the DFP algorithm essentially moves each eigenvalue of HG in turn to unity and so if G is poorly scaled so that, say, all its eigenvalues are large, then HG is liable to become ill-conditioned and DFP can fail. In an attempt to improve the structure of HG he considered updates of the form

$$H^{(i+1)} = \left(H - \frac{Hqq^TH}{q^THq}\right)\gamma^{(i)} + \frac{pp^T}{p^Tq} \quad . \tag{3-43}$$

These are not part of Huang's family because $H^{(i)}$ is multiplied by $\gamma^{(i)}$ but they retain the property $H^{(i+1)}q^{(i)} = p^{(i)}$ and, with exact searches, generate mutually G-orthogonal directions although (3-9) is not satisfied for $j < i$ ($H^{(i+1)}q^{(j)} \propto H^{(i)}q^{(j)}$ is sufficient). Hence $\underline{x}^*$ is still attained within n iterations but $H^{(n)} \neq G^{-1}$. The method has not become widely popular for this reason and because there are other ways of avoiding poor scaling in which case the performance of self-scaling algorithms tends to be inferior to pure BFS. However, the work was well intentioned, the desire being to develop a more reliable algorithm for general functions when line searches are inaccurate. Luenberger felt that the loss of the conjugate gradient property would cause the eigenvalue structure to deteriorate and with it the performance of the minimisation method. In practice, the careful implementation

of a normal BFS algorithm has proved extremely satisfactory suggesting that the difficulties alluded to have been overcome quite effectively by other means.

Before leaving the subject of matrix modification schemes we should remark that our derivation of the DFP update does not coincide with Davidon's[4] line of thought. In fact, he was seemingly unaware of its conjugate direction property and proposed a more complicated algorithm. The initial extension, clarification and simplification of his basic ideas was the work of Fletcher and Powell[3]. Any student of innovative minds is therefore referred to the original paper!

4 FINITE DIFFERENCE DERIVATIVES

4.1 Motivation and efficient implementation

In section 3 it was largely assumed that exact computations could be performed. In particular, we never doubted that the partial derivative vector $\underline{g}$ could be determined precisely. Unfortunately, it is frequently the case that analytic derivatives of F are not available in which case finite difference approximations are normally the best substitute although occasionally the nature of the problem is such that a method which does not involve gradient calculations has to be employed. Our computer programs M0402 and M21UNC assume that finite differences can be used and there is a version of each for use when explicit expressions for $\underline{g}$ can be supplied. Either way, it is assumed that the work involved in evaluating $\underline{g}$ greatly exceeds that required to get F so that $\underline{g}$ is only computed at the end of a line search rather than every time F is calculated. Thus quadratic rather than cubic interpolation is used in the one-dimensional minimisation algorithm.

Now, the efficiency of an optimisation algorithm is judged mainly by the number of function evaluations (NFE) required to solve a particular problem. With finite differences this means that the number of gradient calls has to be kept down and that forward differences (4-1) which involve n further function evaluations are to be preferred to central differences (4-2) which involve $2n$ although the latter are more accurate and will be essential near the solution.

$$g_f = \frac{F(x + \Delta x) - F(x)}{\Delta x} \left[- \frac{\Delta x}{2} \frac{\partial^2 F}{\partial x^2} \right] \qquad (4\text{-}1)$$

$$g_c = \frac{F(x + \Delta x) - F(x - \Delta x)}{2\Delta x} \left[- \frac{\Delta x^2}{6} \frac{\partial^3 F}{\partial x^3} \right] . \qquad (4\text{-}2)$$

Thus for quadratic F central differences are exact except for rounding error. The reader should now begin to appreciate two dilemmas facing us:

(a) Exact line searches waste function evaluations but tend to reduce the number of iterations (*ie* gradient evaluations) needed to solve a particular problem.

(b) Forward differences require only half the function calls of central differences but, being less accurate, tend to increase the number of iterations so that there is usually still a saving but less than a factor of two.

M0402 therefore has the fixed strategy of central differences and a fairly tight line search throughout. M21UNC uses forward differences wherever possible and allows the user to specify the accuracy of the search procedure. Unfortunately, there is no way of automatically deciding how accurate each search should be when $\underline{x}$ is remote from the solution and it becomes a matter for experience to decide whether an overall slack or tight search policy is preferable for a given problem. For most applications such optimal program efficiency will probably not be vital but the opportunity to experiment if desired is at least available in M21UNC.

Before considering the use of finite differences in more detail, we ought to mention the alternatives. In the mid-1960s the intuitive feeling that performing many function evaluations in such close proximity was inefficient contributed to the preferential development of other algorithms. In particular, Powell[17] found a way of generating conjugate directions without using derivatives but the method was very dependent on accurate line searches throughout. A sophisticated simplex method was developed by Nelder and Mead[18] but, like any so-called direct search method, it converged slowly and often failed completely. A deficiency common to both these approaches was their deterioration as n increased and it is now accepted that finite difference derivatives represent the best investment providing that

(i) the objective function is sufficiently smooth;

(ii) the calculation of F is not subject to gross errors (causing $F(x + \Delta x) - \left\{\begin{matrix} F(x) \\ F(x - \Delta x) \end{matrix}\right\}$ to be highly inaccurate),

(iii) the step size (differencing interval), Δx is suitably chosen.

The first successful finite difference implementation was by Stewart[19] although it was later pointed out by Gill and Murray[15] that Δx should remain

132

fixed throughout the minimisation whereas Stewart had allowed it to vary with x. Thus in M0402 and M21UNC the Δx specified by the user is adopted throughout regardless of whether (4-1) or (4-2) is in use. Like the line search accuracy parameter referred to above, this involves a compromise and the opportunity (more important with Δx) for experimentation. There seems to be a case for further research here because the argument for constant Δx is related to the accuracy of $\underline{q}$ (equation (3-2)). One would imagine that not updating H (or B) on an iteration when Δx is changed would be an adequate precuation. Indeed one might also be tempted to consider such a safeguard when switching between (4-1) and (4-2).

In the next two subsections the choice of Δx and criteria governing this switching are discussed.

4.2 Choosing the differencing interval

The essential observation is that the influence of inaccuracy, ε, due to rounding error in the calculation of F increases as Δx decreases while the truncation error of the Taylor expansion increases with increasing Δx. Selecting Δx therefore involves balancing these opposing effects. Note that $\varepsilon > \varepsilon_m$ where ε_m is the precision of computer arithmetic ($\varepsilon_m \doteq 10^{-11}$ for ICL1900 series).

From (4-1)

$$\varepsilon_f^r \doteq \frac{2\varepsilon}{\Delta x} \; ; \; \varepsilon_f^t \doteq \frac{\Delta x}{2}\frac{\partial^2 F}{\partial x^2} \; . \qquad (4\text{-}3)$$

From (4-2)

$$\varepsilon_c^r \doteq \frac{\varepsilon}{\Delta x} \; ; \; \varepsilon_c^t \doteq \frac{\Delta x^2}{6}\frac{\partial^3 F}{\partial x^3} \; . \qquad (4\text{-}4)$$

Choosing Δx to minimise $E = |\varepsilon^r| + |\varepsilon^t|$ gives

$$\Delta x_f \doteq \left(\frac{4\varepsilon}{\partial^2 F/\partial x^2}\right)^{\frac{1}{2}} \; ; \; \Delta x_c \doteq \left(\frac{3\varepsilon}{\partial^3 F/\partial x^3}\right)^{\frac{1}{3}} \qquad (4\text{-}5)$$

and hence the likely error in g when Δx is chosen in this way is

$$E_f \doteq 4\varepsilon/\Delta x_f \; ; \; E_c \doteq 3\varepsilon/2\Delta x_c \; . \qquad (4\text{-}6)$$

132

Relations (4-5) and (4-6) respectively suggest that

$$\Delta x_c > \Delta x_f \; ; \; E_f > E_c \; .$$

For example, taking $\varepsilon = 10^{-9}$; $\frac{\partial^2 F}{\partial x^2} = \frac{\partial^3 F}{\partial x^3} = 1$ gives

$$\Delta x_f = 6.3 \times 10^{-5} = E_f \quad ; \quad \Delta x_c = 1.4 \times 10^{-3} \quad , \quad E_c \doteq 1.0 \times 10^{-6} \; .$$

Although $E_f \gg E_c$ the difference will be unimportant while $g \gg E_f$. The example emphasises the need for central differences near the minimum and gives some insight into how precisely we may expect to locate it. Clearly $g^T g = 0$ will not be reached. Indeed we would be lucky to progress beyond

$$g^T g \doteq \sum_{i=1}^{n} (\varepsilon/\Delta x_i)^2 \qquad (4\text{-}7)$$

and even this might become optimistic if $\partial^3 F/\partial x^3$ is especially large. Unfortunately, we cannot estimate $\partial^3 F/\partial x^3$ but a finite difference approximation to $\partial^2 F/\partial x^2$ is available at no extra cost when $\partial F/\partial x$ is computed by central differences

$$\frac{\partial^2 F}{\partial x^2} = \frac{(F(x + \Delta x) - 2F(x) + F(x - \Delta x)}{(\Delta x)^2} - \frac{\Delta x^2}{12} \frac{\partial^4 F}{\partial x^4} + \ldots \; . \qquad (4\text{-}8)$$

Note that the rounding error is here $\geq \varepsilon/(\Delta x)^2$ but hopefully the relative error will still be small. M21UNC outputs $\partial^2 F/\partial x^2$ as obtained from (4-8) together with the diagonal of $B(\underline{x}^*)$. The two n-vectors should be comparable for the free variables.

It also becomes practical to compute $g^{(0)}$ by central differences and to set up the diagonal elements of $B^{(0)}$ or $H^{(0)}$ from (4-8), providing they are positive, thereby ensuring that B or H at least starts off reasonably well scaled. At the same time one could check via (4-5) that Δx has been reasonably chosen bearing in mind

(i) the cancellation error associated with (4-8) and

(ii) that the ideal Δx for central differences (Δx_c) is probably considerably larger than Δx_f .

Furthermore (as suggested in (4-7)) Δx might need to be different for different variables.

M0402 ignores these possibilities, merely setting $H^{(0)} = I$ and applying the single Δx specified by the user to all variables. M21UNC can establish the elements of $D^{(0)}$ from (4-8) but like M0402 it adopts the same user-specified Δx throughout to generate every element of $\underline{g}$, the rationale being that $(\partial^2 F/\partial x^2)^{(0)}$ may not be representative of later second derivative behaviour and that the user has chosen Δx accordingly. However, Δx could be reviewed periodically (*eg* by inserting a central difference derivative evaluation every few iterations). It might be interesting to implement some of these more sophisticated procedures for Δx to see whether appreciable improvement over the present strategy results. Meanwhile, it is gratifying to observe that the simplified process solves problems quite well although the efficiency can vary considerably with Δx especially, as one might expect, when ϵ is large.

Thus if many similar problems are to be tackled, the user is recommended to rerun M0402 or M21UNC with a variety of Δx to determine the best value. Effectively, he will be discovering ϵ which can be a difficult quantity to estimate *a priori*. Whereas the machine accuracy is clearly the lower limit to ϵ a much higher value might be more realistic depending on the nature and complexity of the process by which F is calculated. Equations (4-5) suggest that on the ICL1906S suitable trial values might be $\Delta x = 10^{-3}, 10^{-4}, 10^{-5}$. It is probably not worth trying to be more precise.

The message of this subsection should by now be fairly clear - that analytic derivatives should be used wherever possible but that numerical approximations will usually be an adequate substitute providing the differencing interval is chosen sensibly and high accuracy in the final solution is not demanded. To this we would add the general advice that when analytic derivatives are programmed they should be checked by differencing before attempting full production.

4.3 Single or central differences?

It was concluded in section 4.2 that single (*ie* forward or backward) differences are more economical overall than central differences but that some means is necessary of sensing when the greater accuracy of the latter becomes vital. In practice two tolerances α_1, α_2, on the line search parameter α are
132 prescribed. Letting $\alpha_1 > \alpha_2 > 0$ we switch to central differences when no lower point having $\alpha > \alpha_1$ can be found after which a line search is deemed to fail if no acceptable $\alpha \geqslant \alpha_2$ is located. Conversely, if when using central differences a step is made such that $\alpha \geqslant \alpha_1$, we switch to (4-1) subsequently.

Plainly then the idea is to use (4-2) when little progress is otherwise being made, indicating either that the minimum is being approached or that the solution path has entered an irregular region.

Thus, in reality the tolerances are based on $||\underline{x}^{(i+1)} - \underline{x}^{(i)}||$. From (3-17)

$$||\underline{x}^{(i+1)} - \underline{x}^{(i)}|| = \alpha||d|| .$$

It is also desirable to take into account the differencing interval Δx because this includes a dependence on the function accuracy ε . The criteria therefore become:

(i) switch between (4-1) and (4-2) at $\alpha_1||d|| = F_1(\Delta x)$;

(ii) deem the line search to have failed if $\alpha||d|| > F_2(\Delta x)$ does not produce a sufficient decrease in F .

Suitable settings are $F_1 = (\Delta x)^{\frac{2}{3}}$; $F_2 = 0.1\ \Delta x$, *ie*

$$\alpha_1 = \frac{(\Delta x)^{\frac{2}{3}}}{||d||} \ ; \ \alpha_2 = \frac{0.1\Delta x}{||d||} . \qquad (4\text{-}9)$$

In addition, it is possible for the error of the forward difference approximation to become large relative to g so strictly speaking it should also be tested, *eg* switch to central differences if

$$\max_{k=1\ldots n} \left(\frac{\frac{2\varepsilon}{\Delta x_k} + \left| \tfrac{1}{2}\Delta x_k \frac{\partial^2 F}{\partial x_k^2} \right|}{|g_k|} \right) > 0.1 . \qquad (4\text{-}10)$$

Now $\partial^2 F/\partial x^2$ cannot be reliably estimated from B except perhaps near x^* while a difference approximation has its own sources of error and involves n extra function calls. Nor can we reasonably assume that the two numerator terms are of comparable size so in practice test (4-10) is not made by M21UNC. However, it does provide another reason (other than to check Δx ; see section 4.2) for occasionally interjecting a central difference iteration.

An important matter arising from (4-9) is the accuracy of the final solution. From the expression for α_2 it is apparent that better than $0.1\Delta x$ should not be expected because the line search will not evaluate F at points any closer together. M21UNC actually replaces Δx in the definition of α_2 by 132

min (Δx,TOLX) where TOLX is the user's accuracy request. However, one would expect that having chosen Δx thoughtfully the user would realise that setting TOLX $\ll \Delta x$ was either unnecessary or unduly optimistic.

At the start of this subsection it was implied that backward differences are sometimes used. Although the subject of bounds on variables will be more fully treated in section 5 it is appropriate to remark here that central difference derivatives are always attempted for variables on bounds. However, it is possible that the overstepping of the bound thereby involved is impermissible in which case a forward or backward difference has to suffice. Similar considerations apply to variables which are free but are within Δx of one or other bound.

4.4 Scaling

Because computer arithmetic has a finite accuracy we cannot guarantee the behaviour of algorithms applied to unscaled problems, *eg* where one variable is say $\sim 10^{-3}$ while another is $\sim 10^{6}$. Clearly, no variable should dominate by virtue of its magnitude and both M0402 and M21UNC require the user to supply a vector of scale factors XS such that $|x_k|/XS_k \sim 1$. This implies that the x_k are not going to change by orders of magnitude between $\underline{x}^{(0)}$ and $\underline{x}^*$. If they do then the user would be advised to transform the variables used in the optimisation (*eg* to $\log x_k$).

Of course, there are other reasons of a more mathematical nature for scaling $\underline{x}$ (and also F). An important one concerns the eigenvalue structure of the Hessian approximation, B. Numerical stability and algorithm efficiency are assisted by B having a small spectral condition number with the eigenvalues spread evenly above and below unity.

Luenberger's self-scaling Hessian updating formula, mentioned in section 3.5 was intended to do this automatically but since it can be accomplished by proper scaling, which is anyway desirable on other grounds, there seems little to be gained especially as the $B \to G$ property is lost.

Note how changing FS, the scale factor for F, will multiply every eigenvalue and every element of g simultaneously

132
$$g_k = \frac{\partial F}{\partial x_k} \times \frac{XS_k}{FS} \quad ; \quad B_{k\ell} = \frac{\partial^2 F}{\partial x_k \partial x_\ell} \times \frac{XS_k XS_\ell}{FS} \quad . \qquad (4\text{-}11)$$

Clearly the g_k can be made arbitrarily large or small by varying the choice of FS in particular, so any convergence criterion based on $g^T g$ assumes sensible scaling. M0402 does not actually test $g^T g$; instead it continues until a line

search fails to find a sufficiently different lower point. It then tries a steepest descent ($\underline{d} = -\underline{g}$) step before terminating. Although the user is called upon to supply a tolerance (ε) its effect on the ultimate accuracy of $\underline{x}^*$ and $\underline{g}^*$ is rather obscure (see section 6.1).

M21UNC does monitor $||\underline{g}||$ so it is particularly important that the scale factors are well chosen. FS does not appear explicitly in M21UNC (or M0402) - it is up to the user to incorporate a suitable (constant) value in his subroutine CALCF. Before exit, M21UNC prints a table to which the user should refer for

(i) verification of the solution;

(ii) sensitivity analysis (see section 8);

(iii) possible revision of $\underline{XS}$ and FS .

In connexion with (iii), the diagonal matrix D is output. Those elements corresponding to bound variables should be 1.0 while the others should be spread above and below unity. If this does not occur then changes to $\underline{XS}$ and FS (using (4-11) with $k = \ell$) might be contemplated although the resulting effects on $\underline{g}$ and $\underline{x}$ must not be forgotten.

With M21UNC the desired accuracy of $\underline{g}^*$ must be specified and the program will terminate when

$$\left.\begin{aligned} &\text{(i)} \quad ||g||_2 = (g^T g)^{\frac{1}{2}} < \text{TOLG} \\ \text{and} \quad &\text{(ii)} \quad ||B^{-1}g||_\infty = \max_k \left|(B^{-1}g)_k\right| < \text{TOLX} \end{aligned}\right\} \qquad (4\text{-}12)$$

or no further progress can be made.

It may be surmised from sections 4.2 and 4.3 that TOLG ~ TOLX ~ Δx is a suitable arrangement but there is no harm in specifying TOLG and TOLX down to $O(\varepsilon_m^{\frac{1}{2}})$. Of course, much greater accuracy can be anticipated when $\underline{g}$ is expressed analytically. Ultimately though, we will always be restricted by the accuracy to which F can be calculated. Examples of potential difficulties are

(i) a differential equation which has to be solved numerically (is the solution/numerical method stable?);

(ii) an iterative process (the termination criteria must be sufficiently stringent);

(iii) decisions (*eg* if $x < 1$ do one thing; if $x \geq 1$ do something else. At least F and $\partial F/\partial x$ must be continuous at $x = 1$).

If the user doubts whether a differencing process can generate a meaningful estimate of $\partial F/\partial x$ then he is going to find it difficult to solve his optimisation problem because any method will ultimately want to make small changes in $\underline{x}$ and will be misled if F is subject to gross errors. An example of a virtually impossible problem is when the calculation of F involves a Monte-Carlo simulation.

However, once a suitable problem is properly posed there should be little difficulty in successfully implementing M0402 or M21UNC especially if $\underline{g}$ can be provided analytically.

5 BOUNDS ON THE VARIABLES

5.1 The philosophies of M0402 and M21UNC

Many practical optimisation problems are constrained only by upper and/or lower bounds on the variables

$$XL_k \leqslant x_k \leqslant XU_k \ . \tag{5-1}$$

For example we might have a set of non-linear equations with several solutions from which we wish to isolate one in particular. This may be achieved by restricting the range of the variables. Following on from this to the over-determined system or nonlinear least squares class of problem where we might be fitting a model to a set of data and know from physical considerations that some or all of the parameters must lie in a certain range. The most common constraint is $x_k \geqslant 0$.

A similar situation occurs when a process is approximated by a formula which is only valid in a certain region of x-space. Bounds can be imposed to prevent the solution path entering the undefined region and possibly terminating at a spurious minimum. Both M0402 and M21UNC require bounds to be specified. The extra labour involved in checking a ridiculous XL or XU (*ie* when there is really no bound) is quite trivial. The two programs have rather different attitudes towards bounds. M0402 adopts the simple strategy of considering every variable all of the time while M21UNC neglects variables on bounds until a minimum on the subspace is neared when the bounded set is searched for a variable to release which will decrease F , *ie* a variable x_k is sought such that

132

$$\left.\begin{aligned} g_k < 0 \quad \text{if} \quad x_k = XL_k \\ g_k > 0 \quad \text{if} \quad x_k = XU_k \end{aligned}\right\} . \tag{5-2}$$

Thus M0402 tends to waste time on every iteration by evaluating partial derivatives which are then not used because they are of the wrong sign. Even if (5-2) is satisfied, zigzagging on and off the bound tends to occur. Convergence is thereby retarded as each iteration consists of only a small step. The potential disadvantage of M21UNC's strategy is that unnecessary effort may seem to be expended minimising on the reduced subspace yet the process is essential to prevent the possibility of returning to that same set of variables and perhaps oscillating indefinitely. In practice M21UNC employs a 'diminishing returns' type of policy whereby variables are considered for release when $\underline{g}^T\underline{g} < \text{TOLG}$ and only if none are suitable does it press on towards the termination criterion $||\underline{g}|| < \text{TOLG}$ (*ie* $\underline{g}^T\underline{g} < (\text{TOLG})^2$). The bounded set is then inspected again and if a variable satisfying (5-2) is discovered we revert to the first criterion and continue with the enlarged subspace otherwise M21UNC is left, providing condition (4-12 (ii)) is satisfied, and the current best point is deemed to be the solution. No difficulties have yet been encountered with this procedure and it of course saves a considerable number of function evaluations by reducing the effective dimensionality of the problem. Indeed, experience with practical problems suggests that once a variable has hit a bound it stays there more often than not. Notice that the above ideas can easily be extended to general linear constraints such that search directions are projected along active constraints and the constraints are held until a minimum is neared when the signs of their Lagrange multipliers, are inspected. For simple bounds λ reduces to g (or $-g$).

It is important that second derivative information relating to variables on bounds is deleted. Thus when x_k reaches a bound the elements of H (likewise for L and D if using B) are reset as follows:

$$H_{k\ell} = H_{\ell k} = 0 , \quad k \neq \ell : \quad H_{kk} = 1 . \qquad (5\text{-}3)$$

This ensures that later on when (5-2) is fulfilled a function decrease will be obtained on the following iteration (because $d_k = - g_k$). Although (5-3) implies discarding of information it is not really a waste because the variable is unlikely to be freed for many iterations during which time no new information regarding the dependence of the function on that variable will be available. 132
Therefore, we could only retain values from just before the bound was reached but their worth when resurrected could not be guaranteed. This is not quite true of M0402 because g_k is evaluated throughout the time that x_k is at a

bound so that the relevant quantities in updates (3-24) or (3-26) could be specified. However, it would be unwise to use the information because it could lead to frequent failure of test (3-25) even with exact searches because $d_k = (-Hg)_k$ does not hold for the bound variables. In other words, G^* might not be positive definite (*eg* if F depends linearly on x_k, $x_k^* = XL_k$ or XU_k necessarily). It only need be positive definite in the subspace of the free variables, a result which is readily extended to general linear and nonlinear constraints such that the Hessian of the Lagrangian function at a constrained minimum has only to be positive definite in the subspace tangent to the active constraint surface.

5.2 An example and a warning

Minimise

$$F(\underline{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \qquad (5\text{-}4)$$

subject to

$$0.2 \leq x_2 \leq 1.0$$

beginning at $\underline{x}^{(0)} = (-1.2, 1.0)$.

The unconstrained solution is $F^* = 0$ at $\underline{x}^* = (1,1)$ but the bounds create other stationary points. Differentiating,

$$g_1 = \frac{\partial F}{\partial x_1} = -400x_1\left(x_2 - x_1^2\right) - 2(1 - x_1)$$

$$g_2 = \frac{\partial F}{\partial x_2} = 200\left(x_2 - x_1^2\right)$$

$$G = \begin{pmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix} .$$

(a) $x_2 = 0.2$ gives $g_1 = 0$ when $x_1 = -0.428, -0.026, 0.454$

(b) $x_2 = 1.0$ gives $g_1 = 0$ when $x_1 = 1.0, -0.995, -0.005$

(a) (i) $x_1 = -0.428$; $g_2 > 0$; $\frac{\partial^2 F}{\partial x_1^2} > 0$ acceptable constrained minimum

(ii) $x_1 = -0.026$; $g_2 > 0$; $\frac{\partial^2 F}{\partial x_1^2} < 0$ unacceptable maximum

132

(iii) $x_1 = 0.454$; $g_2 < 0$; $\frac{\partial^2 F}{\partial x_1^2} > 0$ unacceptable minimum;

(b) (i) $x_1 = 1.0$; $g_2 = 0$; $\frac{\partial^2 F}{\partial x_1^2} > 0$ global minimum

(ii) $x_1 = -0.995$; $g_2 > 0$; $\frac{\partial^2 F}{\partial x_1^2} > 0$ unacceptable minimum

(iii) $x_1 = -0.005$; $g_2 > 0$; $\frac{\partial^2 F}{\partial x_1^2} < 0$ acceptable constrained maximum

where by 'unacceptable' we mean that the bound could profitably be released (by (5-2) or complementary criteria in the case of maxima).

Thus a minimisation can (and both M0402 and M21UNC do) terminate properly at (-0.428, 0.2) rather than (1,1). Now let us consider the Hessian matrix, G . At (-0.428, 0.2) the complete matrix is $\begin{pmatrix} 142 & 171.2 \\ 171.2 & 200 \end{pmatrix}$ which, having a negative determinant, is not positive definite. Hence updating the full B or H approximation to G or G^{-1} while maintaining the positive definite requirement will prevent the solution from being attained. However, if B or H is adjusted as in (5-3) when the bound is first encountered then we would have $B^* = \begin{pmatrix} 142 & 0 \\ 0 & 1 \end{pmatrix}$ which is positive definite. Note that in this example, once x_2 has hit its lower bound, whether or not G is positive definite is determined by the sign of $\partial^2 F/\partial x_1^2$; hence the interpretations given above.

One further point mentioned in section 4.3 is that the specified bounds are not strictly observed in that when evaluating derivatives numerically a bound can be exceeded by an amount Δx . Usually, this is of no consequence but it is not difficult to envisage a disastrous situation (*eg* taking $x^{\frac{1}{2}}$ where $x \geq 0$ supposedly). To guard against this, a facility is provided whereby the user can check for such violation and refuse to calculate F in which case M0402 and M21UNC make do with the appropriate single difference.

5.3 The complete algorithm of subroutine UNCMIN

We are now in a position to sketch the complete algorithm (A-4) which M21UNC implements via subroutine UNCMIN (see also Fig 2).

(1) Initialise L, D . Calculate $F^{(0)}$. Identify any variables on bounds. Set δ = TOLG , $i = 0$.

(2) Compute $\underline{g}^{(i)}$ by central differences.

(3) Compute $\underline{g}^T\underline{g}$. If $\underline{g}^T\underline{g} > \delta$, go to (10).

(4) If forward differences are in use, convert $\underline{g}$ to central differences and go to (3).

(5) Central differences already so test for release of bounds. If a suitable one is found (5-2), set δ = TOLG and go to (10).

(6) No bound variable can be released so if the line search has failed go to (16), otherwise

(7) if $\delta = \text{TOLG}^2$ go to (9).

(8) Set $\delta = \text{TOLG}^2$. If $\underline{g}^T\underline{g} > \delta$, go to (10).

(9) Terminate with IEXIT = 1 if $||B^{-1}\underline{g}||_\infty < \text{TOLX}$.

(10) Output i, NF, F, $\underline{x}$, $\underline{g}$ if required. Set i = i + 1.

(11) Compute $\underline{d}(= B^{-1}\underline{g})$, $\underline{d}^T\underline{d}, \underline{d}^T\underline{g}$. Estimate $\alpha^{(i)}$ from (6-4).

(12) Perform a line search. If unsuccessful, go to (4).

(13) If a new bound hit, reset appropriate elements of L and D to 0 and 1 respectively.

(14) If using central differences, check whether forward differences can be used henceforth ($\alpha^{(i)} > \alpha_1$: see (4-9)).

(15) Compute $\underline{g}^{(i)}$. Update L and D if possible. Go to (3).

(16) Perform local search. If no lower point found, terminate with IEXIT = 4, otherwise go to (2).

If analytic partial derivatives are available then steps (4) and (14) will be omitted and references to central differences are changed to read 'analytic expressions'.

The only aspects of the above algorithm which have not yet been described are the searches (12) and (16). They are therefore the subject of the next section.

6 THE LINE SEARCH AND THE LOCAL SEARCH

6.1 The one-dimensional search

The notion of a line search was introduced in section 3 and refers to the process of selecting $\alpha^{(i)}$ in the quasi-Newton iteration

132

$$\underline{x}^{(i+1)} = \underline{x}^{(i)} - \alpha^{(i)}H^{(i)}\underline{g}^{(i)} .$$

We recall that the rank-1 matrix updating formula does not require a search as such but we still want $F^{(i+1)} < F^{(i)}$ and $H^{(i+1)}$ to be positive definite. The latter proves a stumbling block for the rank-1 correction. Certain rank-2 updates have the property that they can always be applied with safety (except for round-off error) if the line search is exact but it is customary to satisfy a less stringent condition and hope that $p^T q > 0$ is still fulfilled.

The accuracy of the search can be specified in M21UNC (which uses the NPL subroutine LINSCH) but not in M0402. Both subroutines basically use quadratic interpolation to estimate the position of the first minimum along $\underline{d}$. One needs a procedure for determining the initial step $\alpha_0^{(i)}$ after which the minimum can be interpolated or extrapolated, first by making use of the gradient at $\underline{x}^{(i)}$

$$\left.\frac{\partial F}{\partial \alpha}\right|_{\alpha=0} = \underline{g}^{(i)T}\underline{d}^{(i)} .$$

Subsequent quadratic fits are made using three function values. M0402 retains three points such that the minimum is bracketed whereas LINSCH tends to keep the three lowest points and therefore has to include safeguards to prevent absurd extrapolations. In neither routine are partial derivatives required until the line search has been completed.

An important point is that $F^{(i+1)} < F^{(i)}$ is not a satisfactory condition as it stands (Fletcher[8]). Instead, a 'sufficient' decrease is demanded

$$F^{(i+1)} - F^{(i)} < \mu\alpha^{(i)}\underline{g}^{(i)T}\underline{d}^{(i)} \qquad (6\text{-}1)$$

where $0 < \mu < \frac{1}{2}$. LINSCH sets $\mu = 0.0001$. The accuracy of the line search is determined by η $(0 \leqslant \eta < 1)$ which causes $\alpha^{(i)}$ to be chosen so that

$$\frac{F_\sigma - F^{(i+1)}}{\alpha^{(i)} - \sigma} \leqslant -\eta\underline{g}^{(i)T}\underline{d}^{(i)} \qquad (6\text{-}2)$$

where σ is the largest value of α less than $\alpha^{(i)}$ and $F^{(i+1)}$ is the lowest function value found. Thus $\eta = 0$ specifies an exact line search while $\eta = 0.99$ would imply a slack search. Having chosen $\alpha^{(i)}$ in this way it is conceivable that (6-1) will be violated in which case $\alpha^{(i)}$ is successively halved until (6-1) holds. If this happens then the final accepted point will not necessarily be the lowest point found. 132

As was discussed in section 4.3, it must also be ensured that the step $\underline{p}^{(i)} (= \alpha^{(i)}\underline{d}^{(i)})$ is not unrealistic given the accuracy to which F can be calculated. If no satisfactory point is found with $\alpha \geq \alpha_2$ a line search failure is recorded. This normally means that the error in $\underline{g}$ is such that the resulting $\underline{d}$ is an uphill direction. It will be noticed that algorithm (A-4) does not implement the 'soft' lower bound, α_1 such that if no $\alpha \geq \alpha_1 > \alpha_2$ can be found we switch to central differences. No obvious problems have resulted from this decision and it has in general led to greater efficiency.

The one dimensional search thus needs to be a rather sophisticated routine if it is to make the most efficient use of function values. A saving of two or three function calls per line search can result in a considerable reduction in overall running time since objective function evaluation is the dominant process in most practical problems. The choice of η is one best left to the user for it depends on the individual problem and on how derivatives are calculated. It is recommended that one starts with a tight search and then explores the effects of relaxing it (*eg* $\eta = 0.01, 0.1, 0.5, 0.9$).

However efficient the line search procedure may be it will be enhanced by the provision of a good estimate, $\alpha_0^{(i)}$. First, the maximum stop permitted by the bounds on the variables is calculated

$$\alpha_m = \min_k \left\{ \begin{array}{ll} \dfrac{XU_k - x_k^{(i)}}{d_k^{(i)}} & \text{if } d_k^{(i)} > 0 \\ \dfrac{XL_k - x_k^{(i)}}{d_k^{(i)}} & \text{if } d_k^{(i)} < 0 \end{array} \right\} . \qquad (6\text{-}3)$$

We then set

$$\alpha_0^{(i)} = \min \left(1, \frac{2\Delta F}{g^T d}, \alpha_m \right) . \qquad (6\text{-}4)$$

As the minimum is neared and $B \to G$ it is expected that $\alpha \to 1$. The second term on the right of (6-4) is based on a local quadratic approximation to F. After the first iteration we set $\Delta F = F^{(i)} - F^{(i-1)}$. If F were locally quadratic we would have

$$F(x + p) = F(x) + p^T g + \tfrac{1}{2} p^T G p \qquad (6\text{-}5)$$

and

$$g(x + p) = g(x) + Gp . \qquad (6\text{-}6)$$

With an exact search $g(x + p)^T p = g^T p + p^T G p = 0$. Therefore in (6-5) $F(x + p) = F(x) + \frac{1}{2} g^T p$. But $p = \alpha d$, giving

$$\alpha = \frac{2(F(x + p) - F(x))}{g^T d} \quad . \tag{6-7}$$

In (6-4), ΔF is used because $F^{(i+1)}$ is, of course, unknown at the start of the iteration.

M0402 operates slightly differently. It employs the user's tolerance parameter, $E(< 1)$, to define a minimum first step and endeavours to choose $\alpha_0^{(i)}$ to restrict the change in $\underline{x}$

$$\tilde{\alpha} = \min \left(1, \quad \frac{0.1}{||\underline{d}||_\infty} , \quad \alpha_m\right) \tag{6-8}$$

$$\tilde{\alpha}_2 = \frac{0.1E}{||\underline{d}||_\infty} \tag{6-9}$$

then

$$\alpha_0^{(i)} = \max (\tilde{\alpha}, \tilde{\alpha}_2) \quad . \tag{6-10}$$

Steps smaller than $\tilde{\alpha}_2$ are permitted but after two such consecutive steps, M0402 sets $\underline{d} = - \underline{g}$ and terminates if $\alpha < \tilde{\alpha}_2$ again occurs (see the flow chart, Fig 1).

An idea of the intended accuracy of M0402's line search can be obtained from the original matrix update criterion which was much more restrictive than (3-25).

$$|g^{(i+1)T} p| \leq - 0.1 g^{(i)T} p \quad . \tag{6-11}$$

This has now been relaxed to allow more frequent updating. Moreover, the original program reset H to I after every n updates. The current version does not discard H arbitrarily and is consequently more efficient and in practice no less robust.

6.2 The local search

The idea of the local search is due to Gill and Murray and is intended as a means of confirming a solution or for moving away from a saddle point. M0402 does not have such provision and M21UNC only activates it in the event of a line search failure (see algorithm (A-4) at the end of section 5). The procedure as implemented in M21UNC is as follows. 132

(1) Take an arbitrary step away from $\underline{x}$ to $\underline{x}'$.

(2) Choose a direction orthogonal to $(\underline{x}' - \underline{x})$ and take a step to find which way is downhill.

(3) Perform an exact line search along $(\underline{x}' - \underline{x})$ to get $\underline{z}$.

(4) Perform an exact line search along $(\underline{z} - \underline{x})$ to get $\underline{x}''$.

The initial step (1) is of the same magnitude for each free variable and is directed towards the further bound. The step (2) is of the same magnitude but alternate components are reversed to obtain orthogonality. The last component must then be set to zero if there is an odd number of free variables, *eg* if step (1) were $(\psi, \psi, -\psi, \psi, -\psi, -\psi, -\psi)$ then step (2) would be $(-\psi, \psi, \psi, \psi, \psi, -\psi, 0)$. Both line searches can then be performed without violating any bounds because α_m is determined using (6-3) with $d_k = 1$ and ψ is then set by

$$\psi = \min (\text{TOLX}, \tfrac{1}{2}\alpha_m) \qquad (6\text{-}12)$$

where TOLX is the desired accuracy in the x_k . If the second search (4) terminates very near $\underline{x}$ then this is accepted as the best that M21UNC can do and the user is left to ponder why his specified termination criteria have not been achieved. Otherwise, the minimisation continues from $\underline{x}''$. An especial virtue of the local search is that no derivative information is required.

7 M0402 OR M21UNC?

M0402 is five years older than M21UNC, It has therefore been applied more widely and has proved to be a reliable program. Also, it was written entirely in Mathematics and Computation Department and hence may be used by people connected with but outside RAE. M21UNC cannot be distributed elsewhere because of the NPL subroutines within it. It is also a much larger program physically but this should be immaterial in most cases. The main virtues of M21UNC are its improved efficiency and sensitivity information. We have not found any problem on which M0402 performs better (*ie* lower number of function evaluations, NFE) than M21UNC and it is necessarily much less helpful for post-optimal sensitivity analysis (see section 8) and in fact generally less flexible. On the other hand, this makes it relatively simple to use whereas M21UNC does invite the user to experiment with the data which may not always be worthwhile.

132

The data for the two programs (see Appendices B and C for more details) are naturally similar but it is left to the user to arrange its input. Both

need $\underline{x}^{(0)}$, $\underline{XL}$, $\underline{XU}$ and $\underline{XS}$, an upper limit to NFE, a print control parameter, MODE (of operation), Δx (finite difference interval)(if applicable) and tolerance(s) (E for M0402 see section 6.1; TOLG and TOLX for M21UNC). In addition M21UNC needs ETA, the line search accuracy. The user has to provide a subroutine CALCF to evaluate $F(\underline{x})$ and, if analytic derivatives are available, subroutine DERIV is also needed.

Regarding performance, we are not presenting any specific figures because test problems are not very representative of real life although they are helpful during program development. Standard test problems tend to be readily differentiable and the accuracy to which F is calculated is $O(\varepsilon_m)$ because the functions are fairly trivial. It is then not surprising that finite differences work almost as well as analytic expressions. Furthermore, apparently spectacular performance on test problems claimed in the literature is often merely a consequence of the way in which function calls are counted, *eg* an analytic gradient evaluation counted as one or even ignored entirely if $\underline{g}$ is always calculated with F (*ie* a cubic line search). Again, the accuracy of the line search affects NFE and this may well have been optimised or if a slack search is used and gradient calls ignored, NFE will probably be particularly low. However, slack searches are often extremely expensive when finite differences are necessary and this is especially true when minimising penalty functions where $F(\alpha)$ is less likely to be unimodal because of the influence of the constraint terms. Hence the general advice given in section 6.1 to initially use a tight line search when attempting a new problem.

Thus there seems little point in trying to compare performance with other published algorithms but comparison between M0402 and M21UNC is realistic because we know that they have been applied in the same way. When allowance is made for differences in reckoning NFE, M21UNC seems to be as efficient as any program using quadratic line searches described in the literature.

Both M0402 and M21UNC form the basis of nonlinearly constrained optimisation programs. M0402 is used in the SUMT program, M0414K which uses a barrier function to tackle inequality constrained problems while M21UNC appears in M21IPF (Purcell[2]) which implements a variation of Fletcher's[20] 'ideal' penalty function for inequality and equality constraints.

8 SENSITIVITY ANALYSIS

132

Quite often the user will want to know how sensitive is the solution provided by a minimisation routine. For example, he might like to know how flat

the optimum is or what would happen if an active bound were relaxed slightly and F reoptimised while if fitting a curve through a set of points the statistical errors associated with the optimum parameters would be appreciated.

M0402 returns $\hat{\underline{x}}$, $\hat{F}$, $\hat{\underline{g}}$ and $\hat{H}$ but does not provide any other information. It is therefore up to the user to form the relevant expressions below and to interpret them. Such a policy at least requires him to make a conscious effort to extract more details which is perhaps better than merely giving him figures which he may be tempted to employ without properly considering their worth. This will probably be particularly true of problems in which the calculation of F is of low accuracy or when finite difference derivatives have been used (the former usually implies the latter anyway).

Now, M0402 does not aim for any specific accuracy in $\hat{\underline{g}}$ or $\hat{\underline{x}}$ so they must be checked first. $\hat{\underline{g}}$ can be output immediately while, denoting the true solution by $\underline{x}^*$, we have for the free variables

$$\underline{x}^* - \hat{\underline{x}} \doteq - \hat{H}\hat{\underline{g}} \qquad (8\text{-}1)$$

and conditions (5-2) should *not* hold for any bound variable. Note also that the magnitude of derivatives with respect to bound variables indicates the degree to which that bound is restraining F* *ie* relaxing bound k by δx_k will cause F* to decrease by $\approx g_k^* \delta x_k$.

The local curvature of F can be explored only with M21UNC, using either of the columns headed 'Estimates of $\partial^2 F/\partial x^2$. The entries in these columns should be similar for the free variables. One is obtained from $\hat{L}$ and $\hat{D}$ while the other is found from (4-8) (if the finite difference version of the program is being used). $\partial^2 F/\partial x^2$ for the bound variables can also be calculated from (4-8) but 1.0 must necessarily appear in the other column. However, if CALCF prohibits the overstepping of the bound then the entry in the 'Differences' column is set to 1.0E20.

Writing $(0, \ldots 0, \delta x_k, 0 \ldots 0) = (\underline{0}, \delta x_k)$, a Taylor expansion about $\hat{\underline{x}}$ gives

$$F(\hat{\underline{x}} + (\underline{0}, \delta x_k)^T) \doteq F(\hat{\underline{x}}) + \delta x_k \hat{g}_k + \tfrac{1}{2}(\delta x_k)^2 \hat{B}_{kk} . \qquad (8\text{-}2)$$

While (8-2) gives an idea of the degradation in F on changing a variable by δx_k it is not easy to estimate the rise in F* or the changes in the other variables if the system is reoptimised keeping x_k fixed at $\hat{x}_k + \delta x_k$.

132

The difficulty is created by the presence of bounds. To be more precise, although we could with some effort estimate the changes in the free variables we have no way of deciding whether the bounds active at $\underline{x}^*$ would remain active because the necessary second derivative information is not available. Formulae for estimating the new optimum and the corresponding increase in F^* are derived in Appendix A assuming $\hat{\underline{x}} = \underline{x}^*$ and that the set of free variables at the perturbed solution is the same as at $\underline{x}^*$. All things considered, the user will doubtless find it more convenient and meaningful to rerun the program starting at $(\hat{\underline{x}} + (\underline{0}, \delta x_k)^T)$ and feeding in ($\hat{H}$ or) $\hat{L}$ and $\hat{D}$ as the initial (inverse) Hessian. Such a computer run should be relatively short because of the good starting values.

While on this subject, it should be stressed that in principle any positive definite matrix will do to start a minimisation. The unit matrix is chosen by default when no other estimate is available because it is well-conditioned and directs the solution path down the first order steepest descent direction, but clearly for a general problem it will be far removed from $G^{(0)}$ let alone G^*. The finite difference version of M21UNC actually attempts to improve $\underline{d}^{(0)}$ by using (4-8) to generate a better scaled diagonal approximation to $G^{(0)}$.

If the original problem was parameter fitting by nonlinear least squares then the covariance matrix will be of interest. H^* is this matrix except for a multiplying factor. For example, find $\underline{x} = (x_1, \ldots, x_n)$ to minimise

$$F(\underline{x}) = \sum_{j=1}^{m} w_j (y_j - f_j(\underline{x}))^2 ; \qquad m > n .$$

Then an estimate of the residual variance of the best fit is

$$\sigma^2 = \frac{\hat{F}}{m - n}$$

and the covariance matrix is $2\sigma^2 (G^*)^{-1}$.

In substituting $\hat{H}$ or $\hat{B}^{-1}$ for $(G^*)^{-1}$ the user should remember that the degree of similarity is uncertain. However, a higher level of confidence is warranted if analytic derivatives have been used and $||\hat{\underline{g}}||$ is very small. With M21UNC, subroutine HESINV has to be called by the user's program in order to obtain $\hat{B}^{-1}$ explicitly which it does by solving the n simple systems of n linear equations

132

$$\hat{L}\hat{D}\hat{L}^T \underline{h}_k = \underline{I}_k \qquad (8\text{-}3)$$

where $\underline{I}_k$ is the kth column of the unit matrix and $\underline{h}_k$ becomes the kth column of $\hat{B}^{-1}$. Because of the symmetry of $\hat{B}^{-1}$ the k^{th} system need only contain $(n + 1 - k)$ equations.

The various features of M0402 and M21UNC described in this section are summarised in Table 1.

9 CONCLUSIONS

We have attempted to explain the development of quasi-Newton (variable metric) methods for the unconstrained optimisation (minimisation) of nonlinear (objective) functions and to describe in some detail the construction of a FORTRAN computer program, M21UNC incorporating current ideas on efficiency, stability and accuracy.

The corresponding aspects of an older and no less reliable program, M0402, were described simultaneously.

Although some ideas for enhancing M21UNC have been suggested in this Report and are to be investigated, it is unlikely that any dramatic improvement will result. Our primary interest is currently in the area of nonlinearly constrained optimisation where we feel that much remains to be done despite the rapid advances of recent years. Also useful would be a program specially for nonlinear least squares fitting which was more efficient than the general purpose programs described here but equally robust.

Finally, we restate our belief that M0402 and M21UNC are very efficient minimisers of nonlinear functions of several variables subject only to upper and lower bounds. In common with every other implementation they do not claim to be able to locate the global minimum of a non-convex function. If the user is worried that many local minima exist he should try rerunning his program from a different $\underline{x}^{(0)}$. Furthermore, it is recognised that the continuity requirements of the method could rule out a substantial class of problems. At the other extreme, problems in which second derivatives can be calculated analytically are, in our experience, sufficiently rare that a separate program capitalising on the extra information is not justified.

Acknowledgment

M21UNC contains several subroutines written by the Division of Numerical Analysis and Computing at the National Physical Laboratory, Teddington. It is

a pleasure to thank Dr Philip Gill and Dr Walter Murray for them and for the discussions which influenced certain parts of the program developed at RAE.

Appendix A

PERTURBING THE MINIMUM AND REOPTIMISING

Consider the unconstrained problem

$$\text{Minimise } F(\underline{x}) \; ; \; \underline{x} = (x_1, x_2, \ldots, x_n) \; . \tag{A-1}$$

Let $\underline{x}^*$ be a solution, *ie* $\underline{g}(\underline{x}^*) = \underline{0}$.

Then, let one of the variables, say x_n be perturbed by δx_n . To second order,

$$F' = F\left(\underline{x}^* + (\underline{0}, \delta x_n)^T\right) = F^* + \tfrac{1}{2}(\delta x_n)^2 B^*_{nn} \tag{A-2}$$

where $F^* = F(\underline{x}^*)$

and $$B^*_{k\ell} = G^*_{k\ell} = \left.\frac{\partial^2 F}{\partial x_k \partial x_\ell}\right|_{\underline{x}^*} \; .$$

Now, holding the perturbed variable constant, reoptimise F by varying the other $(n - 1)$ parameters. Let the new optimum be $\underline{x}^{**}$ and define $\underline{\delta x}^* = \underline{x}^{**} - \underline{x}^*$.

Then, to second order,

$$F^{**} = F^* + \tfrac{1}{2}(\underline{\delta x}^*)^T B^* \underline{\delta x}^* \tag{A-3}$$

and

$$\underline{g}^{**} = \underline{g}^* + B^* \underline{\delta x}^* = \left(\underline{0}, g_n^{**}{}^T\right) \; . \tag{A-4}$$

Writing

$$B^* = \left(\begin{array}{c|c} \widetilde{B}^* & \underline{b}_n \\ \hline \underline{b}_n^T & B^*_{nn} \end{array}\right) \quad \text{and} \quad \underline{\delta x}^* = \left(\begin{array}{c} \widetilde{\underline{\delta x}}^* \\ \hline \delta x_n \end{array}\right) \tag{A-5}$$

gives, with (A-4),

$$\widetilde{B}^* \widetilde{\underline{\delta x}}^* + \delta x_n \underline{b}_n = \underline{0} \; . \tag{A-6}$$

Therefore,

$$B^* \underline{\delta x}^* = (\underline{0}, \underline{b}_n^T \widetilde{\underline{\delta x}}^* + \delta x_n B^*_{nn})^T \; . \tag{A-7}$$

132 From (A-6)

$$\widetilde{\underline{\delta x}}^* = - \delta x_n (\widetilde{B}^*)^{-1} \underline{b}_n \; . \tag{A-8}$$

(A-7) and (A-8) in (A-3) give

$$F^{**} = F^{*} + \tfrac{1}{2}(\delta x_n)^2\left(B^{*}_{nn} - \underline{b}_n^T(\tilde{B^{*}})^{-1}\underline{b}_n\right) . \qquad (A\text{-}9)$$

Relations (A-8) are therefore estimates of the changes in the other (n - 1) variables when one variable is perturbed and the system reoptimised while expression (A-9) estimates the function value at the new minimum. We would therefore expect the following inequality to hold

$$F^{*} < F^{**} < F' . \qquad (A\text{-}10)$$

$F^{*} < F'$ and $F^{**} < F'$ follow from (A-2) and (A-9) respectively because principal sub-matrices of a positive definite matrix are themselves positive definite. To confirm $F^{*} < F^{**}$ requires the following proposition (see (A-9)) to be verified.

If an arbitrary positive definite matrix B^{*} is partitioned as in (A-5) then

$$B^{*}_{nn} > \underline{b}_n^T(\tilde{B^{*}})^{-1}\underline{b}_n . \qquad (A\text{-}11)$$

Proof

Since B^{*} is positive definite,

$$\underline{u}^T B^{*}\underline{u} > 0 \qquad \text{for all non-zero } \underline{u} . \qquad (A\text{-}12)$$

Let $\underline{u} = (\underline{v}, a)$
where $\underline{v}$ is an arbitrary (n - 1) vector
and a is an arbitrary scalar.

Then (A-12) becomes

$$\underline{v}^T\tilde{B^{*}}\underline{v} + 2a\underline{b}_n^T\underline{v} + a^2 B_{nn} > 0 .$$

Completing the square yields

$$\left(\underline{v} + a(\tilde{B^{*}})^{-1}\underline{b}_n\right)^T\tilde{B^{*}}\left(\underline{v} + a(\tilde{B^{*}})^{-1}\underline{b}_n\right) + a^2\left(B^{*}_{nn} - \underline{b}_n^T(\tilde{B^{*}})^{-1}\underline{b}_n\right) > 0 . \qquad (A\text{-}13)$$

Now the choice $\underline{v} = -a(\tilde{B^{*}})^{-1}\underline{b}_n$ zeroises the first term. Therefore, property (A-12) can only hold if the second term in (A-13) is always positive, which establishes the result (A-11) and hence (A-10).

Clearly, a lot of extra computational effort is involved in calculating $\underset{\sim}{\delta x}^*$ and F^{**} particularly as B^* is not explicitly required for any other purpose let alone inverses of submatrices of B^*. In practice therefore, it is better to rerun the minimisation program as described in section 8.

Appendix B

USER GUIDE TO M0402

B.1 Purpose

To solve the problem

$$\text{Minimise } F(\underline{x})$$

subject to

$$XL_k \leq x_k \leq XU_k \qquad k = 1,2,\ldots,n$$

where F is a differentiable nonlinear function of the variables $\underline{x}$. Finite difference approximations to $\partial F/\partial \underline{x}$ are constructed when necessary if analytic expressions are not available.

B.2 Argument list

CALL M0402 (F,G,N,X,XL,XU,XS,E,MT,IP,NH,H,WR,WD,MODE)

F A REAL variable into which the user may put the value of the objective function corresponding to the initial X (see MODE below). It will contain $F(\underline{x})$ on exit.

G A REAL array of N elements to hold the vector of partial derivatives $\partial F/\partial \underline{x}$. G may be set on entry (see MODE) and will contain $\partial F/\partial \underline{x}$ on exit.

N The number of variables.

X A REAL array of N elements. An initial approximation (unscaled) must be set on entry to M0402 and the best values found will be returned (unscaled) on exit.

XL A REAL array of N elements defining lower bounds on the x_k.

XU A REAL array of N elements defining upper bounds on the x_k.

XS A REAL array of N elements containing positive scale factors for the variables, *ie* $XS_k \sim |x_k|$.

E A REAL variable holding the accuracy parameter. M0402 terminates with MODE = 1 (see below) after three successive iterations in which the largest scaled change in the variables is <0.1E. E = 0.01 is recommended.

MT Maximum number of calls to CALCF permitted. On exit the number of calls actually made is stored in MT.

IP Print control parameter. The intermediate output subroutine, REP is called every IP calls to CALCF (IP > 0) or every -IP iterations (IP < 0). If no printing is required set IP = MT , otherwise IP = - 1 is recommended. On exit, IP indicates the number of iterations performed (i on the flow chart; see Fig 1).

NH Must be set to N*(N + 1)/2 .

H A REAL array of size NH used for storing the lower triangle by columns of the inverse Hessian matrix. An initial positive definite approximation may be supplied, otherwise put H(1) = 0.0 causing MO402 to set the unit matrix in H . The current estimate of G^{-1} is returned on exit.

WR A REAL array of N elements used as working space.

WD A DOUBLE PRECISION array of 3N elements used as working space for the accumulation of matrix-vector products.

MODE On entry; MODE = 1 indicates that only X is supplied.
= 2 indicates that F(X) is also supplied.
= 3 indicates that G(X) is also supplied.

On exit; MODE = 1 convergence criteria satisfied (*ie* no significant further progress can be made).
= 2 F could not be calculated at the point, X , supplied by the user.
= 3 An acceptably low point has been reached.
= 4 The maximum number of calls allowed to CALCF has been reached.

B.3 User subroutines

The calculation of $F(\underline{x})$ must be carried out in a subroutine headed

```
SUBROUTINE CALCF (N,X,F,IC)
DIMENSION X(N) .
```

On entry, IC indicates whether F is required for the line search (IC = 1) or for the estimation of $\frac{\partial F}{\partial x_{(IC-3)}}$ by central differencing. On exit, N and X must be unchanged but the user can set IC = 2 if F cannot be calculated or
32 (if IC = 1) IC = 3 to indicate that the value of F just calculated is low enough to be an adequate solution of the problem. IC should not be altered for any other reasons. The user should so scale his objective function that the F value returned by CALCF is of order unity.

If, on the other hand, analytic derivatives are available, CALCF can only be called with IC = 1 and the user must supply subroutine DERIV to evaluate G.

```
SUBROUTINE DERIV (N,X,F,G)
DIMENSION X(N), G(N)
```

N,X and F should not be altered. F is included in the arguments in case it shortens the calculation of derivatives.

B.4 COMMON area

The subroutine which calls M0402 must include the statement

```
COMMON/M0402A/DELTA
```

where DELTA is the scaled differencing interval ($\Delta x(>0)$) for every variable and should be a data item so that a suitable value can be discovered by experiment. DELTA in the range 0.001 to 0.00001 is suggested (see section 4).

When analytic derivatives are provided via subroutine DERIV, the user should set DELTA = 0.0 .

B.5 Other subroutines

M0402 calls CALCG and REP. CALCG either evaluates derivatives numerically (by calling CALCF 2n times) or, if DELTA = 0 it calls DERIV. The latter course is preferable in every way although the analytic expressions should initially be checked by comparison with the finite difference version. For each call of DERIV, n is added to the count of function calls.

The calling of REP is governed by the parameter IP. It outputs to channel 2 in the following format which the user may alter to suit.

IT NF	iteration number (i); number of function evaluations so far.
F	the objective function $F(\underline{x}^{(i)})$.
X(K),K=1,N	the best (lowest) point so far found.
G(K),K=1,N	partial derivatives $(\partial F/\partial \underline{x})^{(i)}$.

M0402 and CALCG do not produce any output.

B.6 General

(a) M0402 was written by Mr Piggott of Mathematics and Computation Department, RAE with small modifications by the present author. It is a relatively small program and tests on many practical problems have verified its robustness.

(b) Switching between numeric and analytic derivatives merely involves changing DELTA.

(c) There is no restriction on problem size because no arrays are explicitly dimensioned. In general one might expect NFE, the total number of calls to CALCF required to increase at least as fast as n^2 .

(d) For sensitivity analysis see section 8. Regarding accuracy, upon a normal exit (MODE = 1), the user should check that the elements of array G are small ($\ll 1$) for all the free variables and that

$$G_k < 0 \qquad \text{if} \quad x_k = XU_k$$

$$G_k > 0 \qquad \text{if} \quad x_k = XL_k \ .$$

If not, X is not a true minimum and a reason should be sought for the premature exit from M0402.

(e) M0402 can be transported outside RAE for associated applications and implementing it on non-ICL 1900 computers should not present any difficulties.

Appendix C

USER GUIDE TO M21UNC

C.1 Purpose

To solve the problem

$$\text{Minimise} \quad F(\underline{x})$$

subject to

$$XL_k \leq x_k \leq XU_k \qquad k = 1,2,\ldots,n$$

where F is a differentiable nonlinear function of the variables $\underline{x}$. There are two versions of the program, one for when partial derivatives can be expressed analytically while the other uses finite difference approximations. The minimisation algorithm (A-4) (see section 5.3) is implemented in UNCMIN with M21UNC merely acting as an interface with the user's calling segment.

C.2 Argument list

CALL M21UNC (N,X,F,G,XL,XU,XS,JBD,HD,NN,HL,NW,W,HH,TOLX,TOLG,MAXFN,IPRINT, ETA,MODE,IEXIT)

N — the number of variables (≥ 2).

X — A REAL array of size N containing initial values for $\underline{x}$. The solution $\hat{\underline{x}}$ will be returned in X.

F — A REAL variable holding the objective function value. F(X) must be preset when MODE = 2 or 4. On exit, F contains $F(\hat{\underline{x}})$.

G — A REAL array of size N for $\partial F/\partial \underline{x}$. G must be preset when MODE = 2 or 4. On exit, G contains $(\partial F/\partial x)$ for the free variables and $G_k = 0.0$ for the bound variables (see also W below).

XL — A REAL array of N elements containing lower bounds on the x_k.

XU — A REAL array of N elements containing upper bounds on the x_k.

XS — A REAL array of N elements containing positive scale factors for the x_k such that $XS_k \sim |x_k|$.

JBD — An INTEGER array of N elements indicating the state of each variable thus

$$\begin{aligned} JBD(K) = 0 \quad &; \quad XL_k < x_k < XU_k \\ = -1 \quad &; \quad x_k = XL_k \\ = 1 \quad &; \quad x_k = XU_k . \end{aligned}$$

JBD need only be preset correctly when MODE = 4.

HD A REAL array of size N used to hold the diagonal matrix factor, D of the approximate Hessian. All elements of HD must be preset positive when MODE = 3 or 4.

NN Must be set to N*(N - 1)/2 .

HL A REAL array of sixe NN used to hold the unit lower triangular (by rows) matrix factor, L of the approximate Hessian. HL must be preset when MODE = 3 or 4.

NW Must be set to 6*N .

W A REAL array of size NW used as working space. On exit, W is partitioned as follows:

W(1) - W(2N)	not useful
W(2N + 1) - W(3N)	holds $-B^{-1}\underline{g}$ (zero for the bound variables)
W(3N + 1) - W(4N)	holds diag (B) (scaled) computed from L and D .
W(4N + 1) - W(5N)	holds diag (B) (scaled) computed by finite differences (if used).
W(5N + 1) - W(6N)	holds $\underline{g}$ for all variables.

HH A REAL variable containing the scaled step size for finite differencing. It is important that the variables are well scaled (by XS) because HH will be applied to them all. The user should examine a range of values for HH (say 10^{-3}, 10^{-4}, 10^{-5}) as suggested in section 4. HH is not used when subroutine DERIV is available.

TOLX A REAL variable expressing the scaled accuracy required in the variables. UNCMIN attempts to satisfy $\max|(B^{-1}g)_k| <$ TOLX . When using finite difference derivatives there is little point in requesting TOLX $\ll$ HH .

TOLG A REAL variable imposing a tolerance on $\hat{\underline{g}}$. UNCMIN attempts to satisfy $\hat{\underline{g}}^T\hat{\underline{g}} <$ TOLG^2 . Again, setting TOLG $\ll$ HH is probably rather ambitious. With analytic derivatives HH is not needed and smaller values of TOLX and TOLG can be contemplated (say $\sim 10^{-6}$).

MAXFN Maximum number of calls to CALCF permitted.

IPRINT Print control parameter. UNCMIN outputs to channel 2 every IPRINT iterations. IPRINT = 0 suppresses all output. Various messages will
132 appear (see under Output below) when IPRINT $\neq$ 0 to indicate progress or difficulties. The usual setting is expected to be IPRINT = 1 but if the user only wants initial and final values to be output, IPRINT = 1000 is suggested.

ETA A REAL variable specifying the line search accuracy ($0 < \eta < 1$). ETA = 0.01 is recommended initially but many problems will be solved more efficiently by adopting a much larger value (up to say $\eta = 0.9$).

MODE Indicates which quantities are being supplied to M21UNC.

MODE = 0 Only a point X is provided. UNCMIN will begin by computing F and G but will not use equation (4.8) to define the elements of HD . Instead HD(K) = 1 , K = 1,N . This will be the normal setting with the analytic derivative version of the program.

= 1 As MODE = 0 except that equation (4.8) is employed to set up HD . This will be the most common setting with the finite difference program and, of course, cannot be used with analytic derivatives.

= 2 F(X) and G(X) are provided as well. The contents of G for variables initialised on bounds are irrelevant.

= 3 X, HD and HL are supplied. This is the normal setting for sensitivity analysis where HD and HL will have been taken from a previous run.

= 4 X, F, G, JBD, HD and HL are supplied. This option is really intended for constrained minimisation by a sequential penalty function method (*eg* M21IPF see Purcell[2]) but is included here for completeness. On exit, MODE contains the number of function evaluation performed.

IEXIT Indicates the circumstances in which UNCMIN was left.

IEXIT = 1 Normal exit. Convergence criteria satisfied.

= 2 Should not happen unless in MODE = 3 or 4 the user supplies a non-positive definite matrix (*ie* HD contains a negative element).

= 3 MAXFN function calls reached.

= 4 No further progress possible (see under Output below). Normally means that the limiting accuracy has been reached and that to attain the desired accuracy the calculation of F will have to be improved to make $F(\underline{x})$ smoother. Alternatively it may be that a more accurate line search or changes to HH and/or XS will be beneficial.

= 5 Should not occur. It would mean that something has gone wrong in the matrix updating routine MODCHL.

C.3 User subroutines

The calculation of $F(\underline{x})$ must be carried out by a subroutine headed

```
SUBROUTINE CALCF (N,X,F,IC)
DIMENSION X(N)
```

On entry, IC indicates whether F is required for the line search (IC = 1) or for the estimation of $\frac{\partial F}{\partial x_{(IC-3)}}$ by differencing. On exit N and X must be unchanged but the user can set IC = 2 if F cannot be calculated *providing* it is a gradient call (IC $\geq$ 4). Returning IC = 2 to the line search will have unpredictable consequences. The assumption therefore, is that F can be defined everywhere within the specified bounds on the variables but may not be calculable outside (see the end of section 5.2).

If, on the other hand, analytic derivatives are available, CALCF can only be called with IC = 1 and the user must supply subroutine DERIV to evaluate G.

```
SUBROUTINE DERIV (N,X,F,G,JBD,IC)
DIMENSION X(N),G(N),JBD(N)
```

N, X and F should not be altered. F is included in the arguments in case it shortens the calculation of derivatives IC and JBD appear in the arguments to enable DERIV to economise on computation.

IC = 0 Only $\left.\frac{\partial F}{\partial x}\right|_{free}$ are required. The free variables are identified by JBD(K) = 0 .

IC = 1 Only $\left.\frac{\partial F}{\partial x}\right|_{bound}$ are required. The bound variables are identified by JBD(K) = $\pm$ 1 .

If it is more convenient to calculate all derivatives together then, when called with IC = 0 , subroutine DERIV should return IC = - 1 to tell CALCG to retain all elements of G and that a subsequent call to DERIV with IC = 1 would be unnecessary. For the large majority of iterations, derivatives
132 with respect to variables on bounds will be ignored.

C.4 COMMON areas: None.

C.5 Other subroutines

The RAE subroutines are M21UNC, UNCMIN, RESET, HESINV, CALCG and CALFUN. M21UNC is responsible for scaling the variables for UNCMIN and for unscaling quantities on exit. RESET modifies the matrices L and D whenever a bound is encountered. HESINV calculates B^{-1} and therefore an extra $\frac{1}{2}N(N + 1)$ of REAL array space is needed if B^{-1} is specifically required by the user. In the controlling segment,

```
DIMENSION BINV (325)        (ie  N ⩽ 25)
.
.
.
NN = N*(N - 1)/2
NW = 6*N
CALL M21UNC (..............)
.
.
.
NH = NN + N
CALL HESINV (N,HD,NN,HL,NH,BINV,NW,W).
```

Note that HESINV requires 2N REAL locations for working space and that the array W may conveniently be employed because M21UNC does not return anything of significance in its first 2N elements.

CALCG works out the required partial derivatives either by calling the user's DERIV (analytic expressions) or by differencing. In the latter case, depending on how it is called by UNCMIN, CALCG does one of three things.

(a) Evaluates $\partial F/\partial x$ for the free variables (using (4-1) or (4-2) as appropriate).

(b) Evaluates $\partial F/\partial x$ for the bound variables (by (4-2)) until one is found which satisfies (5-2).

(c) Takes backward differences to convert forward differences to central differences for the free variables.

CALFUN interfaces between UNCMIN or LINSCH and the user's CALCF. The function call count is incremented by one every time CALACF is called or by n' every time DERIV is called where $n' = n$, n_f or $n - n_f$ depending on the call and return values of IC . n_f is the number of free variables.

In addition, the following NPL routines are called,

LINSCH To perform the one-dimensional (line) searches.

LDLTSL To solve $B\underline{u} = \underline{v}$ for $\underline{u}$.

MODCHL To add a rank-1 matrix $B^{(i)} \rightarrow \tilde{B}$ (formula (3-39) is actually implemented; $\tilde{B} = B^{(i)} + \frac{\underline{q}\underline{q}^T}{\underline{p}^T\underline{q}}$).

NMDCHL To subtract a rank-1 matrix $\tilde{B} \rightarrow B^{(i+1)}$,
(again with (3-39); $B^{(i+1)} = \tilde{B} + \frac{\underline{g}\underline{g}^T}{\underline{d}^T\underline{g}}$ and the denominator is necessarily negative because $B\underline{d} = -\underline{g}$).

MDCOND To adjust the elements of HD when its condition number becomes too large.

C.6 Output

All printing is controlled by the parameter IPRINT and goes to channel 2.

IPRINT = 0 No output

IPRINT > 0 The following messages can appear

(a) ENTRY TO UNCMIN

(b) CONVERGENCE CRITERION IS NOW: GTG < XXXX

(c) ITERATION XX HESSIAN HAS BEEN MODIFIED BECAUSE COND(D) = XXXX

(d) ITERATION XX FAILURE X (should only get 'X' = 4)

(e) TRY AGAIN WITH CENTRAL DIFFERENCE DERIVS

(f) VARIABLE XX COULD BE FREED

(g) ENTER LOCAL SEARCH PROCEDURE

(a) is printed immediately UNCMIN is entered;

(b) indicates that TOLG has been reduced by UNCMIN because the point supplied already satisfied $\underline{g}^T\underline{g} < (\text{TOLG})^2$.

(c) HD has become ill-conditioned ('XXXX' > 1.0E7);

(d) line search failure. As can be seen from the flow chart (Fig 2) four courses of action are then possible;

(i) if forward differences are presently in use switch to central differences and try again (message (e): can only occur if finite differences are being employed, of course);

(ii) work out derivatives for bound variables looking for one satisfying (5-2).
132 If one is found, message (f) is output;

(iii) perform a local search (message (g));

(iv) when actions (i) to (iii) have been tried in that order without success, UNCMIN terminates with IEXIT = 4.

The summary which UNCMIN provides before exit should be self explanatory (see sample output, Fig 3). The use of it is described in section 8.

In addition, the following quantities are output every IPRINT iterations during a minimisation

IT NF	iteration number (i); number of function evaluations so far.
F	the objective function $F(\underline{x}^{(i)})$.
X(K), K = 1,N	the best (lowest) point so far found.
G(K), K = 1,N	partial derivatives $(\partial F/\partial \underline{x})^{(i)}$. A zero value will normally be output for a variable on a bound.

UNCMIN is the only subroutine capable of output.

C.7 General

(a) M21UNC is more modern and sophisticated than M0402. We believe it to be more efficient, more informative and equally robust.

(b) Two separate programs are provided for numeric and analytic derivatives.

(c) There is no restriction on problem size because no arrays are explicitly dimensioned. In general one might expect NFE, the total number of calls to CALCF required to increase at least as fast as n^2 .

(d) For sensitivity analysis see section 8.

(e) M21UNC must not be supplied to users outside RAE because of the NPL subroutines within it and the conditions under which those routines were acquired.

Table 1

ACCURACY AND SENSITIVITY CHECKS

	M0402	M21UNC
Estimated accuracy of $\hat{\underline{x}}$	$\hat{H}$ and $\hat{\underline{g}}$ are returned	$-\hat{B}^{-1}\hat{\underline{g}}$ is output
Gradient vector, $\hat{\underline{g}}$	Available	Output
Accuracy of $\hat{H}$ or $\hat{B}$	Unknown	Diag (B) is checked by differencing and both vectors output
Restraining effect of bounds	g_k available	g_k output
Flatness of optimum	Unknown	All quantities in (8-2) are available
Perturbing a free variable, fixing and reoptimising	Best to rerun the program	
Covariance matrix for least squares curve fitting	H available	B^{-1} available

LIST OF SYMBOLS

$a^{(i)}, b^{(i)}$	scalars involved in H updating
$B^{(i)}$	the ith approximation to the second derivative (Hessian) matrix
$c^{(i)}, e^{(i)}$	scalars involved in B updating
$\underline{d}^{(i)}$	search direction (= $- H^{(i)} \underline{g}^{(i)}$)
$D^{(i)}$	the diagonal matrix factor of $B^{(i)}$
DELTA	Δx
E	'accuracy' parameter for M0402
ETA	η
$E_{f,c}$	error in the calculation of $\underline{g}$ by forward (central) differences
F	the objective function (= $F(\underline{x})$)
FS	(positive) scale factor for F which the user should incorporate in his subroutine CALCF
$\underline{g}^{(i)}$	the vector of partial derivatives $\partial F/\partial \underline{x}$ at the start of the ith iteration
G	the true second derivative matrix (= $G(\underline{x})$
$H^{(i)}$	the ith approximation to the inverse Hessian matrix
I	the unit diagonal matrix
$L^{(i)}$	the unit lower triangular matrix factor of $B^{(i)}$
n	the number of optimisation variables
NFE	the number of function evaluations (calls to CALCF) required to reach a minimum
$\underline{p}^{(i)}$	$\underline{x}^{(i+1)} - \underline{x}^{(i)}$
$\underline{q}^{(i)}$	$\underline{g}^{(i+1)} - \underline{g}^{(i)}$
TOLG	desired maximum value of $\|\|\underline{g}\|\|_2$ at $\hat{\underline{x}}$ (M21UNC only)
TOLX	desired accuracy of the solution point ($\|\|\hat{\underline{x}} - \underline{x}^*\|\|_\infty$) (M21UNC only)
$\underline{u}$, $\underline{v}$	occasional vectors
$\underline{v}^{(i)}$, $\underline{w}^{(i)}$	vectors involved in B updating
$\underline{x}^{(i)}$	vector of optimisation variables at the start of the ith iteration
$\underline{x}^*$	a true minimum point ($\underline{g} = \underline{0}$)
$\hat{\underline{x}}$	the solution found by the algorithm (hence $\hat{F}$, $\hat{B}$, $\hat{\underline{g}}$ etc)
$\underline{XL}$	lower bounds on $\underline{x}$
$\underline{XS}$	scale factors chosen so that $\|x_k^{(i)}\|/XS_k \sim 1$
$\underline{XU}$	upper bounds on $\underline{x}$
$\underline{y}^{(i)}, \underline{z}^{(i)}$	vectors involved in H updating
$\alpha^{(i)}$	the line search parameter (thus $\underline{p}^{(i)} = \alpha^{(i)} \underline{d}^{(i)}$)
α_m	maximum value of $\alpha^{(i)}$ permitted by $\underline{XL}$ and $\underline{XU}$

LIST OF SYMBOLS (concluded)

$\alpha_1, \alpha_2, \tilde{\alpha}_2$	minimum value of $\alpha^{(i)}$ allowed when using forward (M21UNC), central (M21UNC), central (M0402) difference derivatives
Δx	scaled differencing interval applied to every variable
ε	rounding error in the calculation of $F(\underline{x})$
ε_m	single length floating point precision of the computer
$\varepsilon_{f,c}^{r,t}$	rounding (truncation) error involved in calculating $\underline{g}$ by forward (central) differences
η	line search accuracy parameter; $0 < \eta < 1$ (M21UNC only)

REFERENCES

No.	Author	Title, etc
1	B.A.M. Piggott	Lectures on nonlinear optimisation techniques. Lecture 2. Parameter optimisation without constraints. RAE Technical Memorandum Math 7310 (1973)
2	A.G. Purcell	FORTRAN programs for constrained optimisation. RAE Technical Report (to appear)
3	R. Fletcher M.J.D. Powell	A rapidly convergent descent method for minimisation. *Comput. J.*, 6, 2, 163-168 (1963)
4	W.C. Davidon	Variable metric method for minimisation. AEC Research and Development Report ANL-5990 (1959)
5	D.G. Luenberger	Introduction to linear and nonlinear programming. Reading, Massachusetts, Addison-Wesley (1973)
6	B.A. Murtagh R.W.H. Sargent	A constrained minimisation method with quadratic convergence. In *Optimisation* edited by R. Fletcher, pp 215-246, London and New York, Academic Press (1969)
7	C.G. Broyden	The convergence of a class of double-rank minimisation algorithms. *J. Inst. Maths Applics.*, 6, 76-90, 222-231 (1970)
8	R. Fletcher	A new approach to variable metric algorithms. *Comput. J.*, 13, 317-322 (1970)
9	D.F. Shanno	Conditioning of quasi-Newton methods for function minimisation. *Math. Comput.*, 24, 647-657 (1970)
10	H.Y. Huang	Unified approach to quadratically convergent algorithms for function minimisation. *J. Optim. Theory Applics.*, 5, 405-423 (1970)
11	M.C. Biggs	Minimisation algorithms making use of non-quadratic properties of the objective function. *J. Inst. Maths Applics.*, 8, 315-327 (1972)
12	L.C.W. Dixon	Quasi-Newton algorithms generate identical points. *Math. Prog.*, 2, 383-387 (1972)

REFERENCES (concluded)

No.	Author	Title, etc
13	C.G. Broyden	Quasi-Newton methods and their application to function minimisation. *Math. Comput.*, 21, 368-381 (1967)
14	D.F. Shanno P.C. Kettler	Optimal conditioning of quasi-Newton methods. *Math. Comput.*, 24, 657-665 (1970)
15	P.E. Gill W. Murray	Quasi-Newton methods for unconstrained optimisation. *J. Inst. Maths Applics.*, 9, 91-108 (1972)
16	A.S. Householder	*The theory of matrices in numerical analysis.* p 123, New York, Blaisdell Publishing Co (1964)
17	M.J.D. Powell	An efficient method of finding the minimum of a function of several variables without calculating derivatives. *Comput, J.*, 7, 155-162 (1964)
18	J.A. Nelder R. Mead	A simplex method for function minimisation. *Comput, J.*, 7, 308-313 (1965)
19	G.W. Stewart	A modification of Davidon's minimisation method to accept difference approximation of derivatives. *J. Ass. Comput. Mach.*, 14, 72-83 (1967)
20	R.Fletcher	An ideal penalty function for constrained optimisation. *J. Inst. Maths Applics.*, 15, 319-342 (1975)

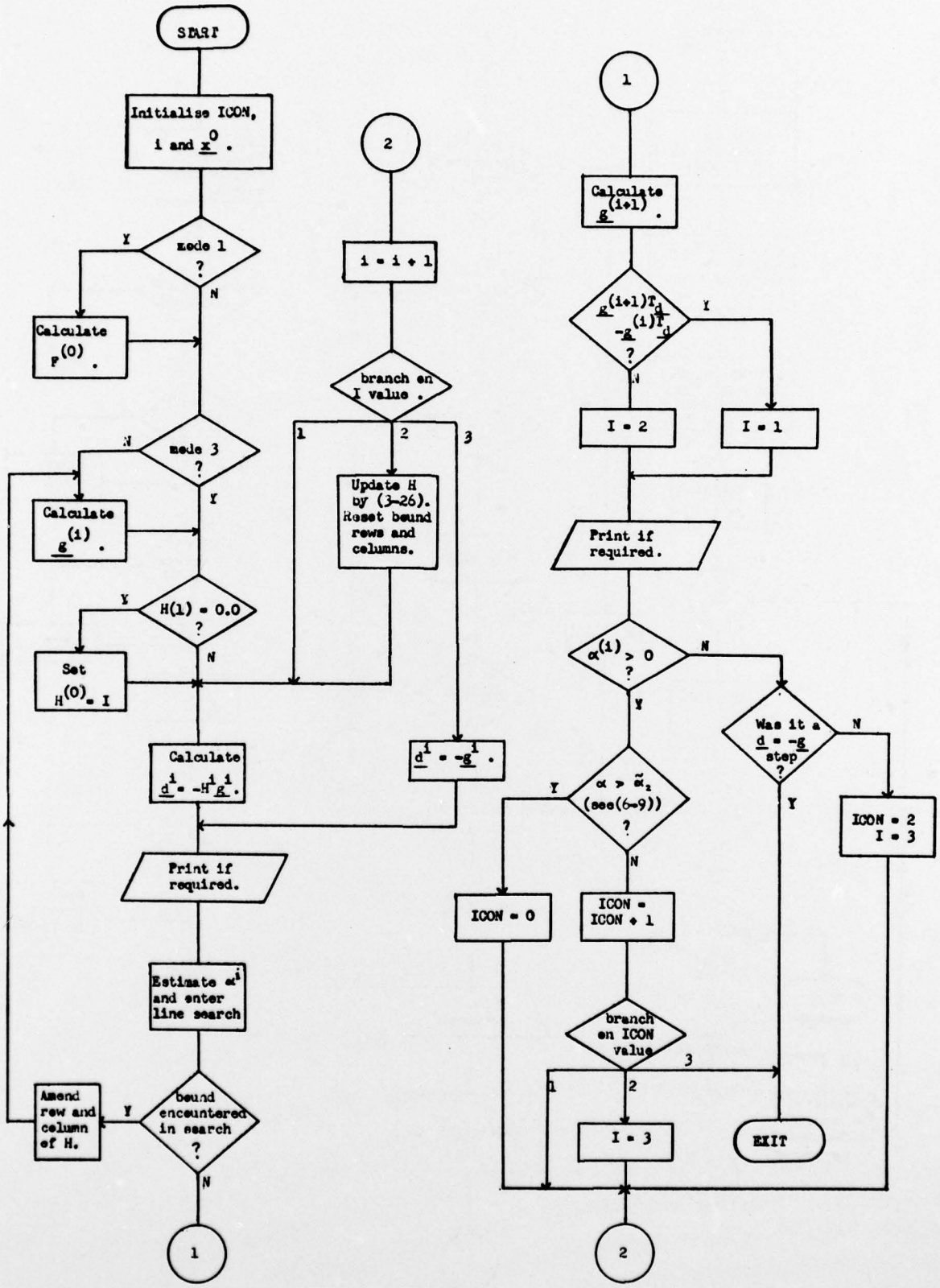

Fig 1 Flowchart for M0402

Fig 2

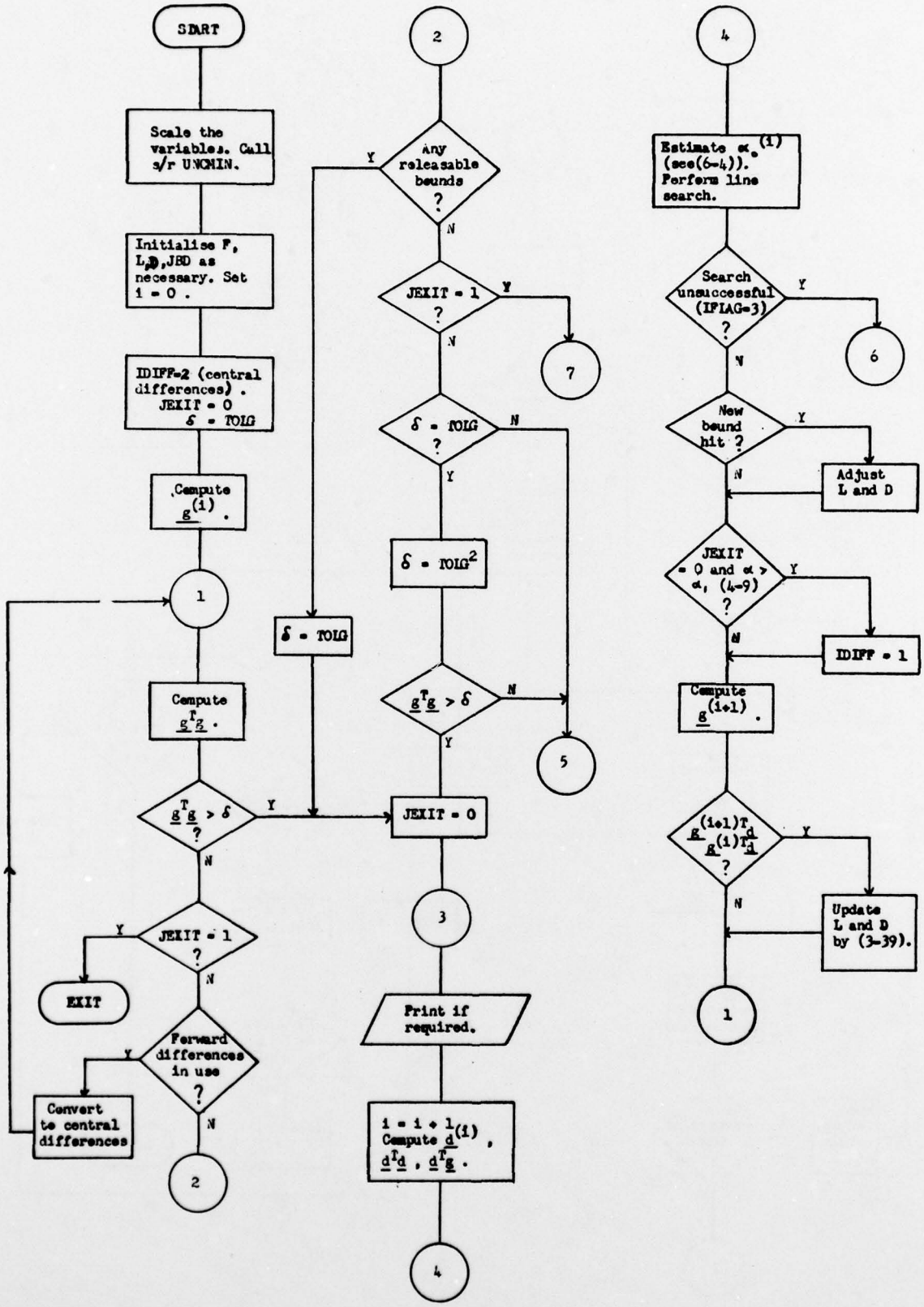

Fig 2 Flowchart for M21UNC

Fig 2 (continued)

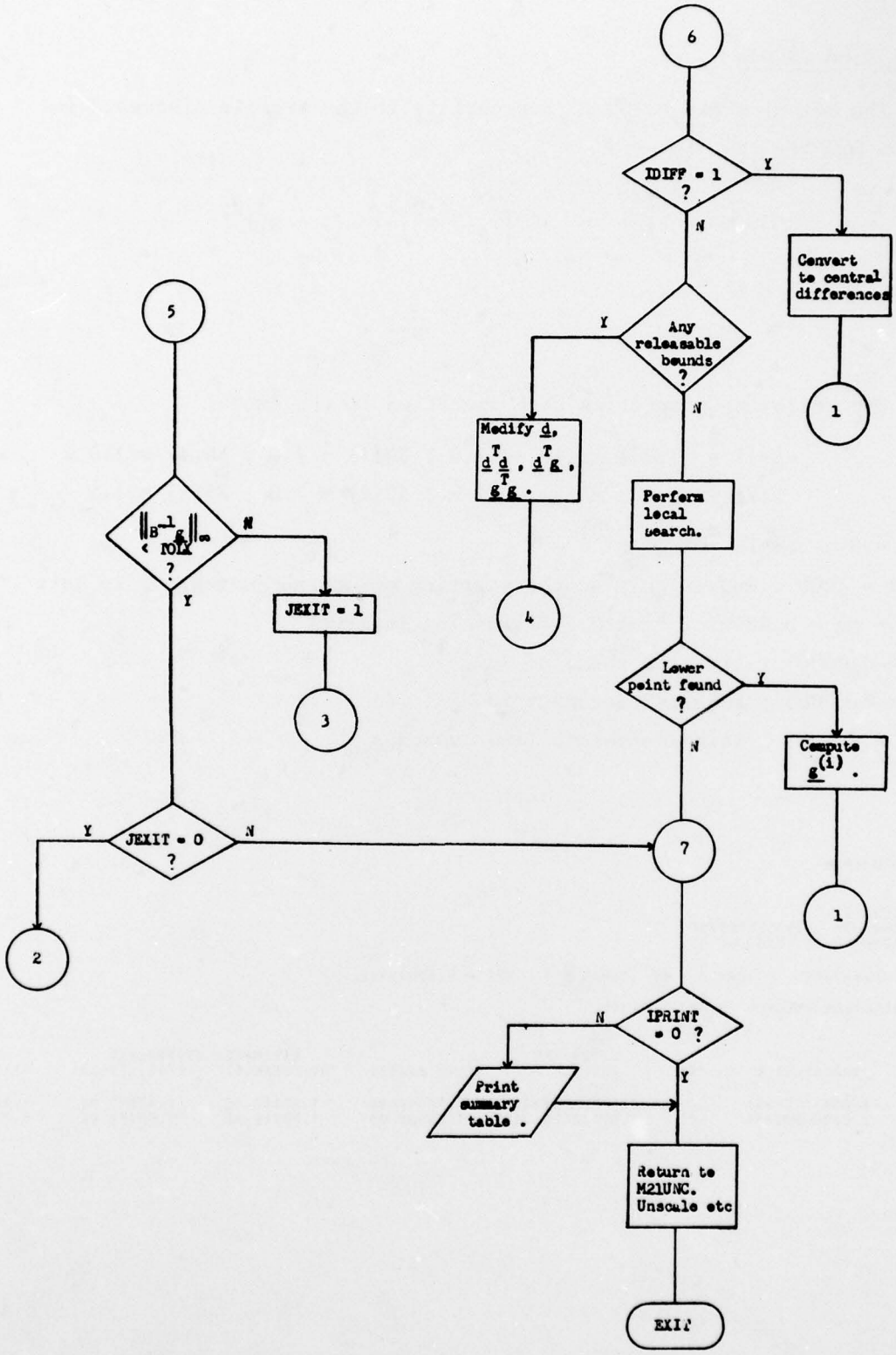

Fig 2 Flowchart for M21UNC (continued)

Output from UNCMIN

The output shown below is appropriate to the example discussed in section 5.2 *ie*

$$\text{Minimise } F(\underline{x}) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

subject to

$$x_2 \geq 0.2 .$$

The following quantities were specified in the data

$$X(1) = -1.2 \; ; \; XL(1) = -2.0 \; ; \; XU(1) = 7.0 \; ; \; XS(1) = 1.0$$
$$X(2) = 1.0 \; ; \; XL(2) = 0.2 \; ; \; XU(2) = 7.0 \; ; \; XS(2) = 1.0$$

MODE = 0 : begin with $B^{(0)} = I$

IPRINT = 1000 : output only at the starting point and just prior to exit

DELTA = Δx = 0.00001 : finite differencing interval

TOLG = 0.00001 ; limiting value of $(\hat{\underline{g}}^T \hat{\underline{g}})^{\frac{1}{2}}$

TOLX = 0.00001 ; required accuracy in $\hat{\underline{x}}$

ETA = η = 0.01 ; fairly accurate line searches

```
ENTRY TO UNCMIN

   0    5
 2.4200000E 01
-1.2000000E 00  1.0000000E 00
-2.1560007E 02 -8.8000030E 01

EXIT ON ITERATION   6  NFE =   57  IEXIT = 1   GTG = 8.4703E-12

LOWEST FUNCTION VALUE =  2.0674599E 00
```

I	VARIABLES X	JBD	ESTIMATED ACC. OF X	G (= DF/DX)	ESTIMATES OF D2F/DX2 DIFFERENCES	DIAG (LDLT)	DIAG (HD)
1	-4.2816139E-01	0	2.0482353E-08	-2.9104E-06	1.4203E 02	1.4209E 02	1.4209E 02
2	2.0000000E-01	-1	0.0000000E 00	3.3356E 00	1.9994E 02	1.0000E 00	1.0000E 00

REPORT DOCUMENTATION PAGE

Overall security classification of this page

UNCLASSIFIED

As far as possible this page should contain only unclassified information. If it is necessary to enter classified information, the box above must be marked to indicate the classification, e.g. Restricted, Confidential or Secret.

1. DRIC Reference (to be added by DRIC)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification/Marking
	RAE TR 77132	N/A	UNCLASSIFIED

5. DRIC Code for Originator	6. Originator (Corporate Author) Name and Location
850100	Royal Aircraft Establishment, Farnborough, Hants, UK
5a. Sponsoring Agency's Code	**6a. Sponsoring Agency (Contract Authority) Name and Location**
N/A	N/A

7. Title: The development of quasi-Newton methods for unconstrained minimisation

7a. (For Translations) Title in Foreign Language

7b. (For Conference Papers) Title, Place and Date of Conference

8. Author 1. Surname, Initials	9a. Author 2	9b. Authors 3, 4	10. Date	Pages	Refs.
Purcell, A.G.			August 1977	66	20

11. Contract Number	12. Period	13. Project	14. Other Reference Nos.
N/A	N/A		Math-Comp 229

15. Distribution statement
(a) Controlled by –
(b) Special limitations (if any) –

16. Descriptors (Keywords) (Descriptors marked * are selected from TEST)

Optimisation.* Minimisation.* Variable metric method. Nonlinear programming.*

17. Abstract

Formulae for updating matrices in connexion with the quasi-Newton iteration are derived so as to emphasise the principles involved. Computational aspects are discussed and two FORTRAN programs for nonlinear minimisation subject to bounds on the variables are described and their use of finite difference derivatives, treatment of bounds, line searches and post-optimal sensitivity facilities are compared to demonstrate the manner in which the subject has progressed in recent years. Brief user guides to the two programs are contained in Appendices.

F5910/1

RAE Form A143